NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A PERFORMANCE ANALYSIS OF AN AD-HOC OCEAN SENSOR NETWORK

by

Kwang Yong Lim

December 2006

Thesis Advisor:	John C. McEachen
Thesis Co-Advisor:	Gurminder Singh

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 2006	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A Performance Analysis of Ad-hoc Ocean Sensor Network	5. FUNDING NUMBERS
6. AUTHOR(S) Kwang Yong Lim	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

This thesis presents the simulation results and performance analysis of IEEE 802.15.4 in an oceanic environment. The 802.15.4 standard allows simple sensors and actuators to co-exist in a single wireless platform. The simulation is performed using Network Simulator, version 2 (NS2) which is an open-source network simulator tool. NS2 is an event driven network simulator developed at UC Berkeley that simulates a variety of networks. Leveraging on the capabilities of NS2, the performance of the IEEE 802.15.4 protocol has been studied based on variations in node density, mobility as well as loading conditions. The mobility model selected for the simulation has considered the ocean effects on the mobile nodes, in particular the surface current. However, the available mobility models (Random Waypoint, Gauss-Markov, Manhattan Grid and Reference Point Group) do not represent the real life mobility in an oceanic environment scenario. As a result, actual data of surface measurement in the Monterey bay area is used to generate the node movements. The results from this analysis provide insights into the performance of IEEE 802.15.4 and its suitability for operating in an oceanic environment.

14. SUBJECT TERMS Ad-hoc networks, Wireless Networks, Mobile Networks, Network Simulation, Performance Evaluation.	15. NUMBER OF PAGES 123
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A PERFORMANCE ANALYSIS OF AD-HOC OCEAN SENSOR NETWORK

Kwang Yong Lim
Civilian, Singapore Technologies Dynamics Pte Ltd
Bachelor of Engineering (Hons), University of Sheffield, United Kingdom, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2006

Author: Kwang Yong Lim

Approved by: John C. McEachen
Thesis Advisor

Gurminder Singh
Co-Advisor

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents the simulation results and performance analysis of IEEE 802.15.4 in an oceanic environment. The 802.15.4 standard allows simple sensors and actuators to co-exist in a single wireless platform. The simulation is performed using Network Simulator, version 2 (NS2) which is an open-source network simulator tool. NS2 is an event driven network simulator developed at UC Berkeley that simulates a variety of networks. Leveraging on the capabilities of NS2, the performance of the IEEE 802.15.4 protocol has been studied based on variations in node density, mobility as well as loading conditions. The mobility model selected for the simulation has considered the ocean effects on the mobile nodes, in particular the surface current. However, the available mobility models (Random Waypoint, Gauss-Markov, Manhattan Grid and Reference Point Group) do not represent the real life mobility in an oceanic environment scenario. As a result, actual data of surface measurement in the Monterey bay area is used to generate the node movements. The results from this analysis provide insights into the performance of IEEE 802.15.4 and its suitability for operating in an oceanic environment.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude to those who have lent their support in my master study.

Firstly, to my beloved wife, Fern, who has continuously encourage and been most supportive in my studies. And for her dedication and patience which have been the driving forces of my determination and achievement.

My appreciation to both my advisors, Professor John McEachen and Professor Gurminder Singh, in giving me advices and guidance for the completion of this thesis.

Also, to senior lecturer Guest Arlene for her patience in teaching and advising on the movement patterns of Oceanography.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. GENERAL

Mobile Ad-hoc NETworks (MANETs) advocate self-configuration and adaptation of wireless links between communication devices (mobile nodes) that operate autonomously or as an extension to a wired infrastructure. The wireless communication links that form the network have to be characterized by a dynamic topology with no fixed infrastructure since the nodes are highly mobile. The nodes are equipped with antennas for reception and transmission within the communication medium through broadcasting. These nodes within MANETs have salient properties including limited power, relatively short communication distance, low processing power and limited bandwidth.

Advancements in sensor technology have made MANETs a viable solution for many commercial as well as military applications. But there has been relatively limited focus on using MANETs in an ocean environment. A typical scenario would consist of a collection of nodes, which are ocean buoys containing sensors that are partially submerged at various positions within a designated operating area. These nodes would be at various depths and will be drifting slowly towards as well as apart from each other. An example military application of such a system in an ocean environment would be for quick deployment for early warning defense. Wireless networks and large-scale sensors provide information and sensing dominance for network centric warfare. Such a system needs to be robust for frequent and unpredictable changes in topology due to the mobility of ocean buoys. It also has to be able to handle large variations in information load with the possibility of information transfer being multi-hopped without centralized control.

B. MOTIVATION

Conventional protocols waste energy in collisions and idle listening. IEEE 802.15.4 is a standard that has been designed for low data rate, low power consumption and low cost which is ideal for MANET applications. There are a host of technical difficulties for implementing this technology for oceanic applications. An oceanic system creates a challenging environment where the ocean current and waves are dynamic and unpredictable which is highly variable as a function of time and space. These factors would cause interference and bit errors at the physical level resulting in unreliable and intermittent connectivity and hence hinder the scalability of protocols. This study focuses on performing the necessary theoretical research to conduct analysis of the performance of IEEE 802.15.4 in an ocean environment using an open-source network simulator tool.

C. OBJECTIVE AND SCOPE

The objective of this thesis is to perform the necessary research to conduct analysis on the performance of IEEE 802.15.4 in an oceanic environment using an open-source tool called network simulator, version 2 (NS2). A simulation model will be built and the various performance behaviors will be investigated that are relevant to a generic wireless networks (such as packet delivery ratio, delivery latency, control overhead and transmission collision) as well as behaviors that are specific to MANETs (such as association, orphaning and transmission methods).

The experiments for performance analysis are simulated using the latest version of NS2 2.29 [1]. NS2 is a discrete event simulator that is targeted for networking research purposes. It provides substantial support for simulation of protocols over wired and wireless networks. In this thesis, besides the installation of NS2, an IEEE 802.15.4 extension [2] (contributed code) will be utilized. This code conforms to IEEE P802.15.4/a18 draft based on the physical layer and the MAC

layer behavior of a wireless network. For the performance analysis, an appropriate mobility model representing the oceanic environment has to be selected. In essence, this thesis endeavors to answer the following questions:

1. Are the available mobility models in NS2 suitable to simulate the ocean environment? Currently there are several models that have been used for research investigation. A few of the models considered are Random Waypoint, Gauss-Markov, Manhattan Grid and Reference Point Group mobility model. Each of these models will be examined in details for suitability to represent node movement in an ocean. If it is determined that these models are not suitable, what are good alternatives to simulate the ocean mobility model?

2. With the selected mobility model, how does IEEE 802.15.4 perform in the oceanic environment? Simulations will be conducted using NS2 and evaluated against a set of performance metrics.

D. THESIS OUTLINE

This chapter provides the introduction and motivation to the study of IEEE 802.15.4 protocol. Chapter II introduces the theoretical background to this research. An overview of IEEE 802.15.4 as well as key concepts behind the ocean effects on mobile nodes is presented. Also, mobility models that are specific to the simulation are discussed. These mobility models are broadly classified as independent or group-based. Chapter III will specifically focus on the methodology in computing node movements from measured data of the Monterey bay. The computed movement will be used to generate the scenario file in NS2. Chapter IV is dedicated to the simulation environment and setup with Chapter V discussing the results of the simulation. Finally, Chapter VI will conclude these studies and recommend further actions and propose future studies.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THEORETICAL BACKGROUND

This chapter provides an introduction to IEEE 802.15.4 as well as key concepts and theory relating to this research. The various mobility models used in simulating the MANET are discussed. Since the mobility models are determined by the environment, factors that contribute to the movement of nodes are studied, especially the ocean surface. These ocean effects are taken into consideration for the selection of mobility models, if applicable. Finally, the performance matrices used in the evaluation of IEEE 802.15.4 for ocean environment are detailed.

A. IEEE 802.15.4

The technology for wireless communications has made tremendous advances, in which most attention has been given to wireless local area networks (WLANs) and technologies involving the IEEE 802.11 standard. While IEEE 802.11 standard focuses on high-end wireless communication system, the IEEE 802.15.4 standard is designed for low cost, low power applications with less stringent requirements on throughput and latency (see Figure 1). The primary objective is to provide reliable data transfer with reasonable battery life for short range operations while maintaining simple and flexible protocol.

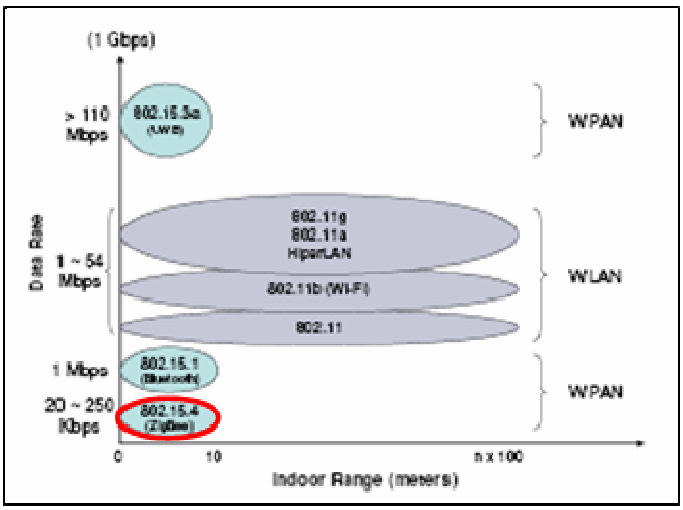


Figure 1 – Wireless Networking (from ref. [3])

1. OVERVIEW OF IEEE 802.15.4

The IEEE 802.15.4 standard defines the physical layer (PHY) and medium access control (MAC) sub-layer specifications for Low Rate - Wireless Personal Area Networks (LR-WPANs). Currently, the development of the upper layers is led by the ZigBee™ Alliance through definition of application profiles utilizing the simplified five layer ISO/OSI reference model shown in Figure 2. The IEEE standards board approved the IEEE P802.15.4B (D6) Draft Revision [4] in June of 2006, which defines the detailed specifications. Unlike WLANs having higher operating range and data-rate, WPANs are designed to function in a personal operating space which can be extended up to 10m omni-directional. Within a LR-WPAN, a device can be assigned to either a 16 bit short or a 64 bit extended address during association. Hence, a single network could potentially accommodate 65,536 (2^{16}) devices.

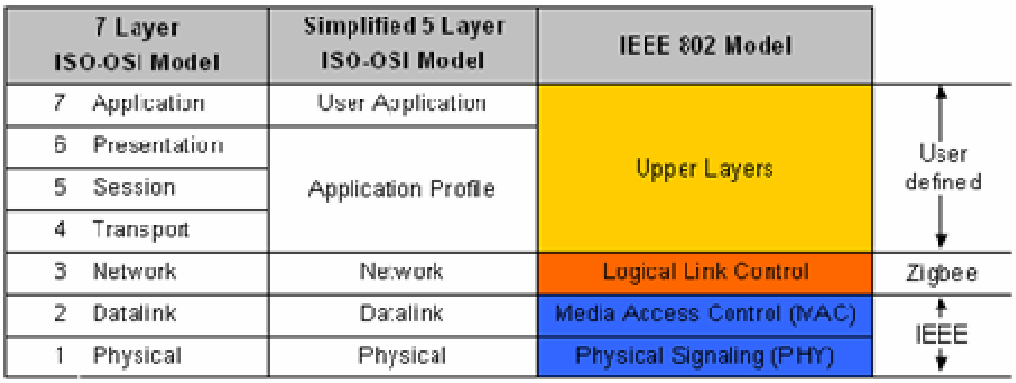


Figure 2 – ISO-OSI Reference Model and IEEE 802 Model (from ref. [6])

Some of the unique characteristics of a LR-WPAN are:

- Over-the-air data rates of 250 kb/s, 40 kb/s and 20 kb/s.
- Star or peer to peer operation.
- Allocated 16 bit short or 64 bits extended addresses.
- Allocation of guaranteed time sots (GTSs).
- Carrier sense multiple access with collision avoidance (CSMA-CA).
- Full acknowledgement protocol for transfer reliability.
- Low power consumption.

- Energy detection (ED).
- Link quality indication (LQI).
- 16 channels in the 2450 MHz band, 10 channels in the 915 MHz band and 1 channel in the 868 MHz band.

The following sections highlight some important design features [4, 5 & 6] of this standard.

2. Network Topologies and Device Architecture

Devices that participate in a LR-WPAN can be of 2 types, a full function device (FFD) or a reduced function device (RFD). Each RFD can only associate with a single FFD at an instance whereas FFDs can communicate with other FFDs or RFDs. A FFD contains the complete set of MAC services and is able to operate as a network coordinator or a network device. On the contrary, a RFD is simply a network device with a reduced set of MAC services and usually used for simple applications. Three types of topologies are supported, namely star, peer-peer and cluster tree as shown in Figure 3. The decision on which topology to adopt is dependant on the operating distance, typically one-hop star and multi-hop peer-to-peer or cluster tree topology when communication range exceeds 10m.

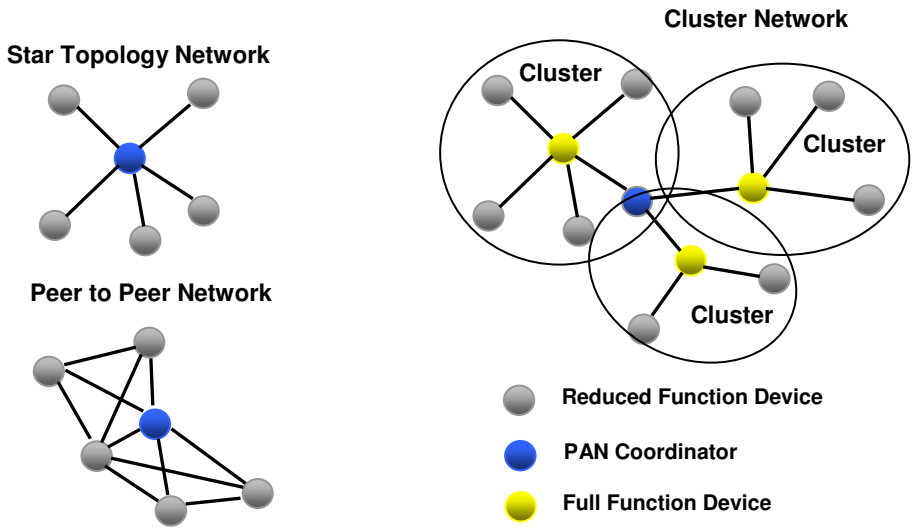


Figure 3 – IEEE 802.15.4 Network Topologies (from ref. [6])

With star topology, communication is established between devices by a single central controller known as the PAN coordinator. The PAN coordinator (which is a FFD) acts as a hub that forms direct links to other devices. These devices, consisting of FFDs or RFDs, form around the PAN coordinator and act as data terminal locations. In the peer-to-peer topology, each device in the network can communicate with any devices within its radio range. Although a PAN coordinator is required somewhere in the network, each device can form multiple links to other devices, hence increasing redundancy at the expense of complexity. The cluster tree is similar except that the network hierarchy is more complex. This topology simplifies routing and reduces direct links at the expense of data traffic latency.

The architecture of a LR-WPAN device, shown in figure 4, comprises the PHY layer and MAC sub-layer. The PHY layer consists of an RF transceiver including a low level control mechanism while the MAC sub-layer provisions for access to the physical channel for all transfer types. Through the service specific convergence sub-layer (SSCS) an IEEE 802.2 logical link control (LLC) can accesses the MAC sub-layer. Each layer will contain services which have the capability to build functions on the services of lower layers and present them to the next higher layer. Every event will require passing of service primitives from a layer to another through a service access point. There are 14 PHY and 35 MAC primitives described in IEEE 802.15.4.

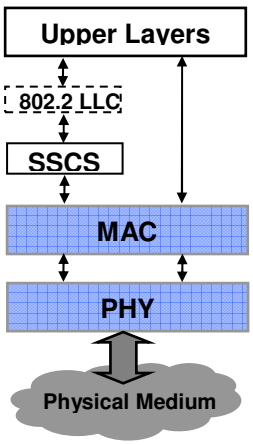


Figure 4 – LR-WPAN Device Architecture (from ref. [6])

3. Functional Features

In the IEEE 802.15.4 standard, some of the functional features used are the superframe structure, different data transmission methods, low power consumption as well as self-configuration and orphaning.

- Superframe structure. The superframe (shown in Figure 5) is a special network communication architecture introduced for time division multiplexing. The superframe consists of an active and an inactive period where beacons in a beacon-enabled network are periodically transmitted for synchronization. Within the active portion, devices in the network can transmit with slotted Carrier Sense Multiple Access – Collision Avoidance (CSMA-CA) in the contention access period (CAP). In the contention free period, only devices with a guaranteed time slot (GTS) can perform data transmission. In CSMA-CA, transmitting devices will listen to the channel for a predetermined time to check for activities on the channel. If the channel is idle, the device is permitted to transmit and if busy, the station has to delay its transmission.

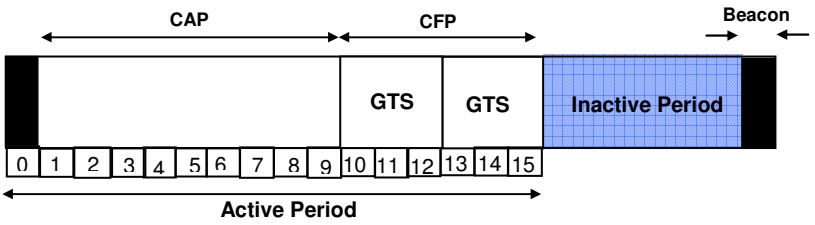


Figure 5 – A Typical Superframe Structure (from ref. [3])

- Data transmission methods. Data transfer can occur either from a device to a coordinator, coordinator to a device or from one peer to another in a peer-to-peer multi-hop network. In general, the data transfer process can be classified into direct transmission, indirect transmission and GTS transmission. Direct data transmission will happen for the three data transfer cases and depending on whether the network is beacon-

enabled or beaconless, a slotted or unslotted CSMA-CA is used. Indirect data transmission applies only to data transfer from a coordinator to its devices where the coordinator keeps a data frame in a transaction list and waits for the device to extract data. Devices will check on beacon frames received from the coordinator to determine if data packets are pending in the transaction list. Also, it is possible, but unlikely that indirect data transmission occurs in beaconless mode where unslotted or slotted CSMA-CA is used in the data extraction procedure. Finally, GTS data transfer occurs both from a device to its coordinator and from coordinator to devices. GTS transmission does not require the use of CSMA-CA protocol.

- Power Consumption. Due to their mobile nature, devices are usually battery operated and hence power consumption is a vital factor. Several power-saving mechanisms are incorporated in IEEE 802.15.4 to prolong device battery life. Most of these mechanisms operate in beacon enabled mode where the receiver of coordinator or its devices are disabled for a specific period during direct transmission. Devices can also be configured to low power (i.e. sleep) state when there is no data transmission or transmitting GTS data transfer at low duty cycle. Other inherent properties to reduce power consumption are the small CSMA-CA back-off period and short transceiver warm-up time.

- Self-configuration and Orphaning. The self-configuration capability is part of the association and disassociation functions in the MAC sub-layer. This capability will automatically establish a star network and also perform self configuration of peer-to-peer network. Orphaning mechanism provides detection of link or node failures for devices in beacon-enable mode and is tracking beacons. When a device misses a pre-determined number of beacons consecutively, the device will perform a re-alignment procedure to establish link with its coordinator.

4. PHY Layer

One of the primary reasons for the failure of several proprietary technologies is the inability to adapt. However, IEEE 802.15.4 has been designed to be applied worldwide supporting three different operating frequency bands, with a different number of supporting channels and at specific data rates (Table 1).

Frequency band	# of channels	Data rate (Kbps)	Applicability	Restrictions
2.4 GHz	**16**	**250**	**Worldwide**	**Unlicensed**
915 MHz	**10**	**40**	**USA**	**Licensed**
868 MHz	**1**	**20**	**Europe**	**Licensed**

Table 1. IEEE 802.15.4 Operating Frequencies.

There are 2 types of frequency bands defined in the PHY layer, the license-free industrial scientific medical (ISM) 2.4 GHz band and the licensed 868/915 MHz band. In total, there 27 channels and 3 different data-rates that is specific to the frequency band. The availability of the ISM band eliminates the need for a license and allows development of devices to be used anywhere in the world. With that, investment risk is greatly reduced which results in lower cost devices. Despite its simplicity, IEEE 802.15.4 also strives to provide flexibility which allows selection of operating in any one of the channels. The basis for the selection is usually dependent on availability, congestion state and data rate of each channel (in terms energy and cost efficiency).

The PHY layer acts as the interface between the MAC sub-layer and the physical medium and provides both data services and management services accessed through two service access points. The following tasks are the responsibility of the PHY layer.

- Activation and deactivation of radio transceiver. Depending on the requests from the MAC sub-layer, the internal operating state of the transceiver can be de-activated (sleep mode), transmitting or receiving.

- Energy detection (ED) within the current channel. This service performs the estimation of the received power within the channel bandwidth and this result can be used in the network layer for channel selection or for the purpose of clear channel assessment (CCA).
- Link quality indication (LQI) for received packets. This measurement is derived from receiver ED estimates, signal-to-noise ratio, or both measures to compute the link quality or strength where packets are received. The result of this measure is used in the network or application layer.
- Clear channel assessment (CCA) for CSMA-CA. CCA is performed by the PHY layer using at least one of three modes of energy, carrier sense and carrier sense with energy. In energy mode, a medium is reported busy if any detected energy is above the ED threshold. As for carrier sense mode, a medium is reported busy when the detected signal has the modulation and spreading characteristics of IEEE 802.15.4. The last mode will report a busy medium when both above-mentioned conditions are met.
- Channel frequency selection. Upon receiving request from MAC sub-layer, the PHY layer will tune to the selected channel.
- Data transmission and reception. This is the primary task of the PHY layer where various modulation and spreading techniques are employed depending on the frequency band.

5. MAC Sub-layer

The MAC sub-layer acts as the interface between the SSCS and the PHY layer, providing MAC data and management services. The features of MAC sub-layer includes beacon, network configuration, channel access, guaranteed time slot (GTS) and link management. The following tasks are the responsibility of the MAC layer.

- Beacon generation for PAN coordinator and synchronization. A FFD that is designated as a coordinator has the ability to operate in beaconless or beacon-enabled mode. Beacon-enabled mode allows the use of superframe that has a structure which is bounded by network beacons and is divided 16 equal sized slots. Beacons are transmitted periodically for describing the superframe structure, synchronization of attached device and identifying the network. Synchronization is required for data polling, energy saving and detection of orphaning.
- Association and disassociation of network. MAC sub-layer association and disassociation supports the automatic setup and self-configuration of star and peer-to-peer network.
- Employing CSMA-CA mechanism for channel access. In consideration of the low data rate in LR-WPAN, the request-to-send (RTS) and clear-to-send (CTS) mechanism are not utilized in the CSMA-CA.
- Allocation and management of guaranteed time slot (GTS) mechanism. While in beacon-enabled mode, GTS allows allocation of dedicated resource to a particular device within a certain portion of the superframe.
- Maintain reliable link between two MAC sub-layer entities. Various mechanisms such as CSMA-CA, CRC data verification, frame acknowledgement and retransmission, are employed by the MAC layer to provision for reliable link.

B. MOBILE AD HOC ROUTING PROTOCOLS

Due to the highly dynamic nature of mobile nodes and the absence of a central controller, traditional routing protocols used for a wired network cannot be applied directly to a MANET. Some of the considerations required in the design of MANET routing protocols include the mobility of nodes, unstable channel states and resource constraints such as power and bandwidth. In a MANET, the movement of nodes will cause communication between nodes to be disrupted

from frequent path breaks and reconnections. Also, the broadcasting of radio channels can be highly unstable and the network layer has to interact with the MAC layer for available channels. In addition, power availability is often limited since the nodes are connected to batteries. Currently, the network and upper layers of IEEE 802.15.4 is being implemented by the ZigBee Alliance. Details of Zigbee Routing protocols will be discussed and other ad hoc routing protocols will not be addressed in this thesis.

1. Zigbee Routing Protocols

The MAC and PHY layers of IEEE 802.15.4 have no notion of node resources and energy. At the network layer, Zigbee supports node activation, network discovery, network formation and routing of packets. Similar to IEEE 802.15.4, Zigbee is an industrial standard that is targeted for low data rate, low power consumption, low cost as well as long life wireless network operations. Zigbee employs a basic master-slave configuration where each master can support up to 254 slaves that can be nested to support a hierarchical routing strategy [7]. In addition, Zigbee provides a unique feature of redundancy in communication hence eliminating single point of failure in mesh networks. The routing algorithms defined for use by Zigbee are the Ad Hoc On Demand Distance Vector (AODV) protocol and the Cluster Tree protocol. Detailed Documentation for these two protocols is provided in the Zigbee specification [8].

2. Ad Hoc On Demand Distance Vector (AODV) Protocol

The AODV is a pure on-demand acquisition algorithm system that allows message passing through neighboring nodes to nodes that are unreachable [9 & 10]. Routing is accomplished by discovering the shortest possible route and ensuring that these routes do not contain loops. AODV also has the ability to adapt to route changes and can create new routes when error occurs. Individual mobile nodes operate as a special router where routes are obtained only when required with minimal reliance on periodic updates. Nodes not within the active

paths neither maintain any routing information nor participate in exchanging routing tables. However, a route needs to be discovered or maintained when there is communication between two nodes or that the nodes are in the direct path between the source and destination nodes. In essence, AODV seeks to minimize the number of control messages sent.

There are several techniques employed for a mobile node to be aware of other nodes in its vicinity, and one such technique is local broadcasting of “hello” messages. Routing tables of the neighboring nodes are organized to have optimized response time for local movements and establishing request of new routes. The primary objectives of the algorithm are [6]:

- Broadcast discovery packets when necessary
- Distinguish between local connectivity management and general topology maintenance.
- Disseminate to neighboring mobile nodes that are likely to need the information about changes in local connectivity.

For the instance where a destination or intermediate node has no routing information, the Path Discovery process is initiated. Each node keeps track of neighboring nodes by listening for the “hello” messages that each node broadcasts at set intervals and by also maintaining two separate counters, the sequence number and broadcast id. The process begins with the source node initiating route request (RREQ) packets to the neighboring nodes. The source address and broadcast id will uniquely identify the RREQ. The broadcast id is incremented for a new RREQ. When an intermediate node receives a RREQ, it increases the hop count and rebroadcasts the RREQ to other neighboring nodes. If that particular node has received a similar RREQ, the redundant RREQ will be dropped. Along the path from the source to destination node, a reverse path is automatically set up where every node maintains the record of the neighboring node that sends the broadcast.

In summary, the route to the destination is determined by comparing both destination sequence number in the node's route entry and the RREQ. The node will respond by either rebroadcasts of the RREQ or unicasts a route reply (RREP) packet back to the node in which request is received. When the RREP packet routes back to the source, a forward pointer is established on each node and timeout information is updated. This process will establish a uni-directional route and is repeated in the reverse direction for a bi-directional route. Due to movement of nodes, there could be errors during routing of messages. A route error message (RERR) allows nodes to adjust routes by removing all routes containing bad nodes from the routing table. This specification is detailed in RFC 3561 [11]. The path discovery process, forward and reverse path are shown in Figure 6.

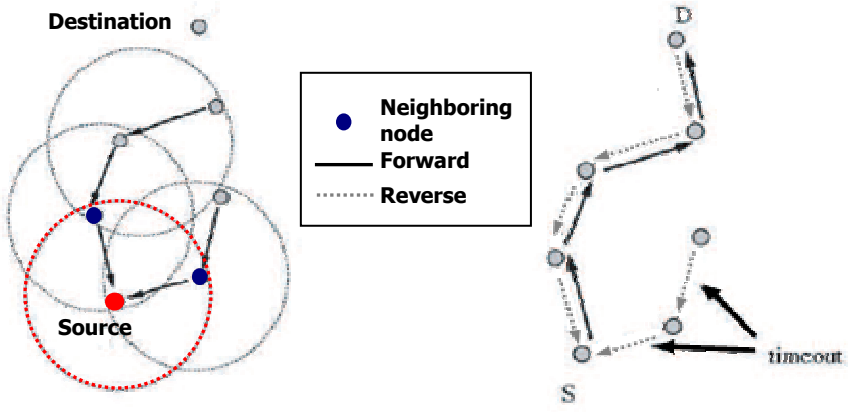


Figure 6 – Path Discovery, Forward and Reverse Path (from ref. [10])

3. Cluster Tree Protocol

The Cluster Tree Protocol utilizes link-state packets to form a single cluster network or a potentially large cluster tree network. The network is mainly self-organized and network redundancy is maintained to achieve a high degree of fault resistance and self-repair. Essentially, nodes form a cluster during the self-organizing process and these formed clusters will connect to each other. Within a cluster of nodes, a cluster head is selected which will assign a unique

node ID to each individual node member during the cluster formation process. The connection between clusters is established using the Designated Device (DD) which assigns a unique cluster ID to each cluster. Basically, the DD is a node having high computing ability and large memory space. A typical cluster tree network is shown in Figure 7. Below sub-section details the process for single and multi cluster network.

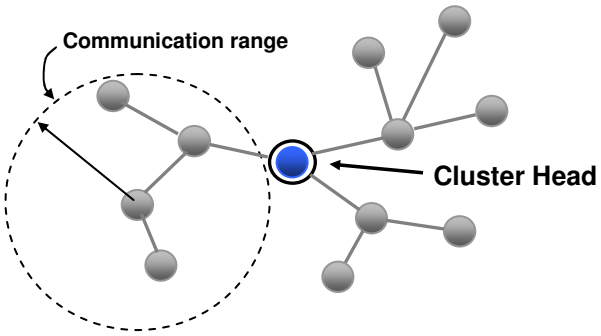


Figure 7 – Typical Cluster Tree Network

a. Single Cluster Network

The process of cluster formation begins with the selection of cluster head, followed by the cluster head expanding links to other node members forming a cluster. There are several ways to select the cluster head, based on the initiating node or stored parameters. When a particular node is activated, that node will listen for “hello” messages (i.e. beacons in the IEEE 802.15.4 MAC layer) from other nodes. After a certain duration when no messages are received, the node appoints itself as the cluster head and sends out “hello” messages. If no connection request is received, the node reverts to its original status. For the other option, the selection is determined from stored parameters such as transmission range, power capacity, computing information or location information.

As the cluster head, the node will periodically broadcast “hello” messages which consist of a partial MAC address and node ID 0. Neighboring nodes will send a

“connection request” message upon receiving the “hello” message. The cluster head node will in turn reply with a “connection response” message containing the node ID and the respective node member will acknowledge the receipt of message. Due to the dynamic mobile nature, all member nodes within the transmission range of cluster head have a star topology with one hop. This cluster can also expand to a multi-hop structure with each node having multiple connections. For the instance where all node IDs have been used or other limits have been reached, connection requests will be rejected by assigning a unique node ID. Also, if no update is received from a member node, it is eliminated from the cluster.

It is possible for a node to receive a “hello” message from another cluster. The receiving node will include the cluster ID (CID) of the transmitting node and will update the cluster head through a link state report. This process allows the cluster head to be aware of the entire topology for better optimization. If there is a need to change the topology, the cluster head will send a “topology update” message. In addition, the periodic “link state report” message is used to indicate the presence of a troubled node. A cluster head in trouble will result in stoppage of the “hello” messages and member nodes will be re-configured.

b. Multi-Cluster Network

A network having multi-clusters (as shown in Figure 8) will need a Designated Device (DD) for assigning a unique cluster ID to individual cluster heads. The DD also serves to calculate the shortest route to clusters and keep nodes within the network informed. DD is usually the cluster head of cluster 0 and begins by sending “hello” message. Neighboring cluster heads respond by sending a “connection request” message to join cluster 0 followed by a request for a CID to the DD. When a cluster head is assigned to a new CID, its member nodes are informed using the “hello” message. If a node member receives the “hello” message from the DD, CID0 is added to the neighboring list and updates

its cluster. This node will then be selected as the border node for that particular cluster and the “network connection” message is sent to establish a connection to the DD resulting in the border node being a member node of cluster 0. A “CID request” message is then sent and upon receiving a “CID response” message, a “network connection response” message containing a new CID is sent to the parent cluster head. Members in the cluster will be informed through “hello” messages. Other clusters not in the vicinity of the DD utilize intermediate clusters to obtain the CID. Eventually, the DD should gather the entire tree structure of the clusters. Individual nodes belonging to a cluster have to maintain a record of their parent and child clusters as well as the border node ID associated with the clusters.

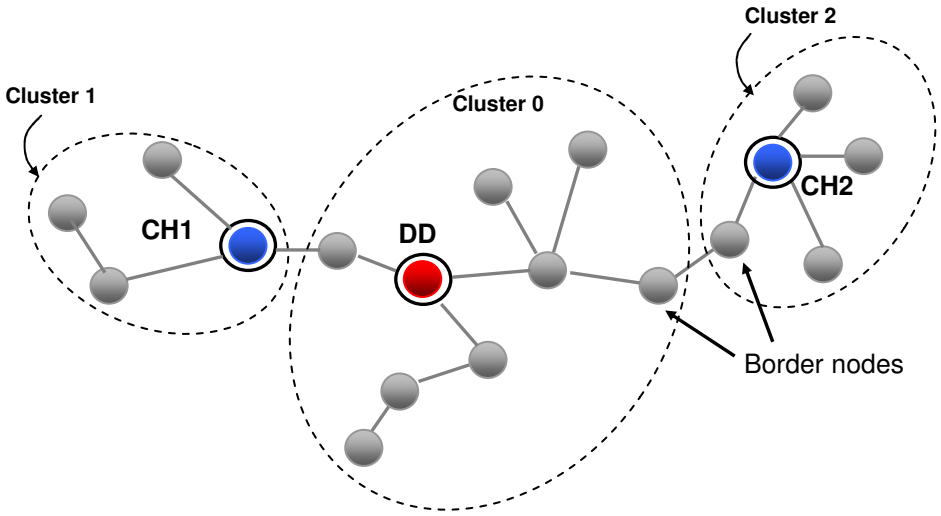


Figure 8 – Typical Multi-Cluster Network

Periodic “network link state report” messages are sent by cluster heads to report link state information to the DD for computing route optimization and to update network redundancy. In return, results of the updated route are sent through a “topology update” message from the DD to the cluster. To prevent network downtime caused by the DD, a backup DD is often assigned. Border routers connect the clusters by performing relaying functions where messages are forwarded to border nodes of an adjacent cluster or a destination node that resides in the cluster. Only the DD can communicate with all nodes in the

network and the broadcast message is forwarded by the border node from the parent to the child cluster.

C. OCEAN EFFECTS ON MOBILE NODES

When nodes (devices on a buoy) are deployed in an oceanic environment, there are various factors that affect their movement. However, the ocean surface current and tidal waves have the most effect on the mobility of the nodes.

1. Ocean Surface Current

An ocean current can be described as a directed movement of oceanic water at the ocean's surface. It can be transient, affecting a small area, or essentially permanent, extending over a long horizontal distance [12]. Currents exist at all depths in the ocean. In some regions, two or more currents may flow in different directions at different depths. Hence, ocean currents can be generally characterized into two types, surface and sub-surface. Essentially, the current system is complex with the actual current flow varying both spatially and temporally. The driving force for ocean currents comes from the atmosphere (mainly from winds), resulting in generation of surface circulation patterns. Such pattern is caused by the interaction of Coriolis deflection, wind drag and the pressure gradient. Wind is air moving across the Earth's surface and is generated by asymmetrical heating of Earth by the sun. Although the wind strongly affects the surface layer, its influence on the ocean is restricted to about 100 meters (325 feet) in depth [13 & 14]. Since this thesis is only concerned with surface currents which cause movement of the floating nodes, the sub-surface layer will not be considered.

a. Coriolis Deflection

Coriolis deflection is caused by the rotation of the Earth due to decreasing velocity at Earth's surface with increasing latitude. The deflection

results in the surface current moving in a clockwise direction (bending to the right) in the Northern Hemisphere and counter-clockwise in the Southern Hemisphere (bending to the left). Therefore, the water currents will flow down the pressure gradient at an angle instead of directly towards the low pressure.

b. Wind Drag

Air at the poles is cooled and hence sinks, creating a region of high pressure at the surface. Differential heating on the Earth's surface by the sun causes air to rise and creates a region of low pressure at the surface of the equator. These differences in air pressure will give rise to winds. Surface winds blow in a regular flow pattern and are influenced by the differential heating of air across Earth's surface and Coriolis deflection. As the wind blows across the ocean surface, the air molecules that are dragged along the surface collide with the surface water molecules and transfer energy through frictional drag. This process results in momentum being transmitted from the air to the water molecules, setting the surface water in motion.

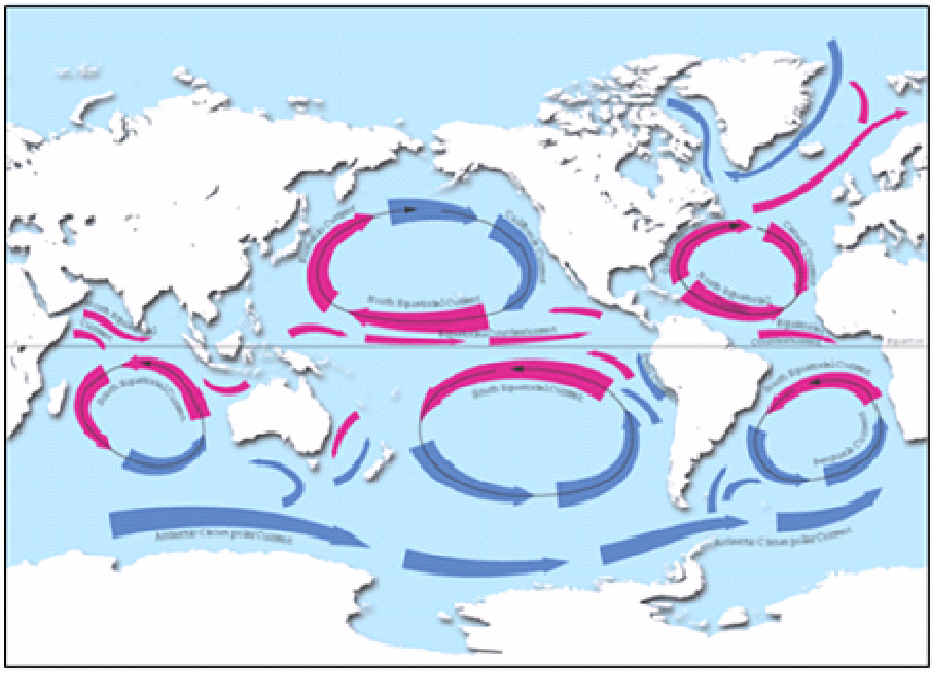


Figure 9 – Surface Ocean Currents (from ref. [15])

Due to the process being inefficient, the speed of the generated current is only approximately 3% of the wind speed. As such, the surface current resulting from wind drag is very much affected by surface winds. Eventually, currents come into contact with continents and are deflected, creating current loops. This phenomenon of creating current loops is known as circulation gyres and is shown in Figure 9.

c. Pressure Gradients

A pressure gradient is a change of pressure across a horizontal distance. With a greater differential pressure over a specific distance, the pressure gradient will increase. Ideally, pressure gradient is almost always balanced by Coriolis force and hence their net result causes no movement. However, the sea surface is warped into broad mounds and depressions. Mounds are caused by convergence (i.e. water flows together and sinks) while depressions are caused by divergence (i.e. water rises to surface and flows outwards). With the variation in the height on the surface, the pressure gradient will rise. When water is flowing down pressure gradients (high to low pressure) on the ocean's irregular surface, it is deflected (a function of location and speed) by Coriolis. This forms a sea surface topography affecting surface circulation.

d. Ekman Spiral

While the surface water moves approximately 45° to the right of the wind in the northern hemisphere, the net transport of water is approximately 90° off of the wind, and a phenomenon called Ekman Transport occurs. With depth, the speed is reduced and the direction of the current changes. This flow pattern is known as the Ekman spiral, as shown in Figure 10.

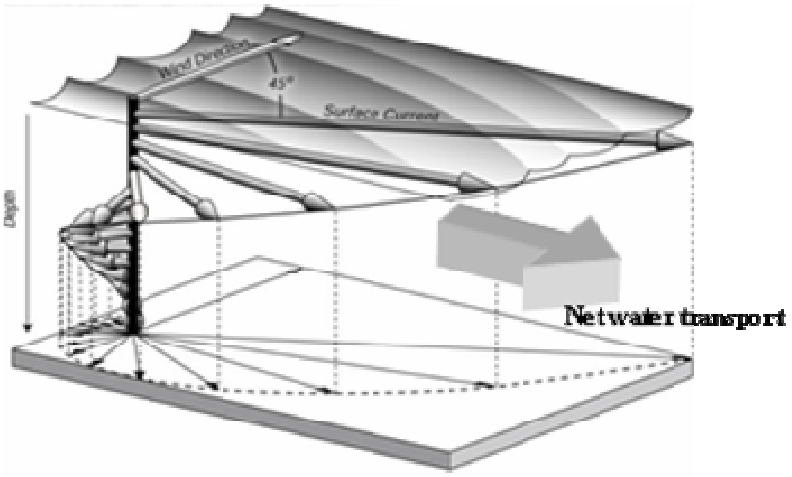


Figure 10 – Ekman Spiral in Northern Hemisphere (from ref. [16])

Surface currents and the Ekman Transport result in the process of upwelling or downwelling (see Figure 11) which is an important aspect of effects along the coasts. When water moves to the right of wind caused by Ekman Transport, upwelling occurs. This movement of water offshore is replaced by cold water moving from below to the surface. Similarly, movement of water to the right of the wind direction caused by Ekman transport will results in downwelling. Water moved offshore is replaced by waters from below.

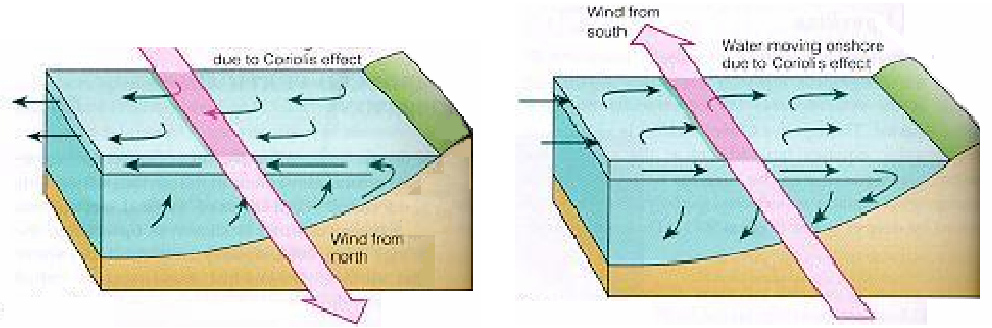


Figure 11 - Upwelling Effect and Downwelling Effect (from ref. [17])

2. Tidal Current

Tides are essentially very long-period waves that cause the sea level to rise and fall in response to the forces exerted by the moon and sun [13 & 18].

Tides are phenomena caused by the gravitational pull of the sun and moon, and counterbalancing that force is inertia. Effectively, gravity and inertia will cause two major tidal bulges [19]. One of the tidal bulges is from the lunar gravitational attraction that “draws” the ocean to the side nearer to the moon. Another is on the opposite side where the inertial force exceeds the gravitational force due to less gravitational attraction from the moon and hence forms the other bulge. The size and position of the bulges are very much affected by the sun where there is a complex interaction between these forces and the lunar forces.

From the horizontal movement caused by the rising and falling of tides, water currents are generated. There are two kinds of flow from this tidal current, flood flow and ebb flow. Flood flow occurs when the tidal current is coming towards the coast or high tide is ensuing (see Figure 12). Ebb flow is where tidal current is receding to the sea or low tide ensuing. The strongest currents (flood or ebb) usually occur before or near the time of the high and low tides. The weakest currents occur between flood and ebb currents, and are known as slack tides. In the open ocean, tidal currents are relatively weak. The speed of tidal currents can reach up to several kilometers per hour near coastal regions.

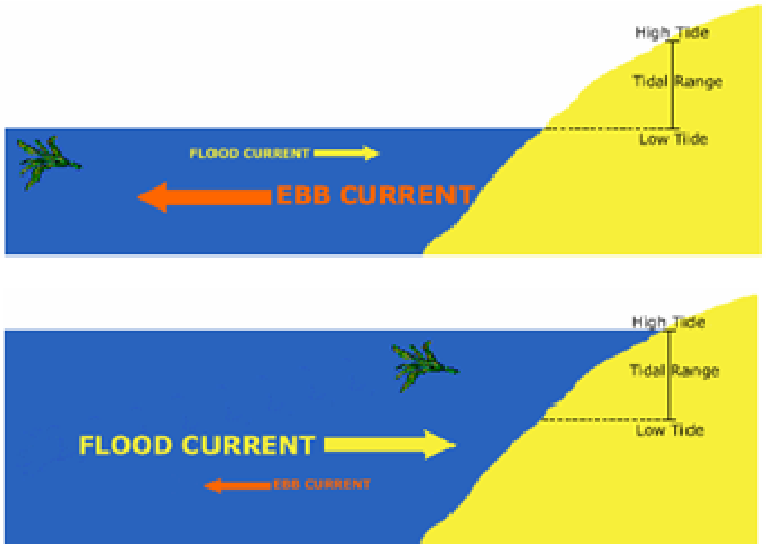


Figure 12 – Ebb current at low tide and Flood current at high tide (from ref. [20])

D. MOBILITY MODELS

In order to simulate IEEE 802.15.4 protocol for an ad-hoc network, it is imperative to use a mobility model that accurately represents the ocean environment. The mobility model should attempt to represent the movement of actual mobile nodes floating on the sea surface. Changes in speed and heading must occur within reasonable time steps.

1. Overview of Mobility Models

Previous work [21] has shown that the chosen models have a great impact on the results of the simulation and hence the evaluation of the IEEE 802.15.4 protocol. Essentially, such models are categorized into those that are related to cellular networks or ad-hoc networks. For this thesis, the main focus is on ad-hoc wireless networks where nodes can move freely within an area of influence. A route must be established between the intermediary nodes for communication between a pair of nodes. Hence, the modeling of the node positions is vital as performance is dependant on the transmission range which is fairly short with respect to the size of the field. Existing mobility models used in simulations can be divided into two distinct classes, independent or group-based. In the latter category, the nodes exhibit a certain relationship between each other and that relationship will determine the movements within the field. Independent models, on the contrary, model the movement of each node independently from other nodes.

2. Mobility Models in BonnMotion

Mobility models are used in NS2 to generate the scenario and such scenario can be generated by BonnMotion software [22]. BonnMotion software is a Java program developed within the Communication Systems group at the

Institute of Computer Science IV of the University of Bonn, Germany. It is capable of creating and also providing analysis of mobility scenarios. There are currently five mobility models, and scenarios generated can then be exported for NS2. The models [23,24] available in BonnMotion are the Random Waypoint model, two versions of Gauss-Markov model, Manhattan Grid model and Reference Point Group Mobility model. An overview of each model is provided below.

a. Random Waypoint Mobility Model

The Random Waypoint Mobility (RWM) model is a simplistic model to represent randomness in movement patterns. The required parameters are the pause time between changes and the direction and/or speed during each change. The nodes in this model will appear to move randomly within the confinement of the simulation boundary. Initially, nodes are placed at certain locations in the simulation area. During the simulation run, each node will remain stationary for the pause time duration. Upon the expiry of pause time, nodes will move toward a new arbitrary position determined by a randomly selected speed, uniformly distributed between a minimum and maximum speed. These nodes will remain in the new position until the pause time has reached and the process repeats until the end of simulation. Figure 13 below shows a scenario generated using RWM model.

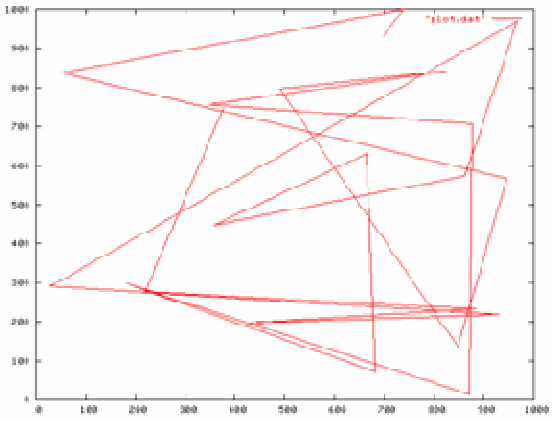


Figure 13 – RWM model scenario (from ref. [25])

b. Gauss-Markov Mobility Models

Gauss-Markov Mobility Model is designed to adapt various degree of randomness in the mobility pattern with the use of one tuning parameter. There are two models implemented, the original Gauss-Markov model and Gauss-Markov model. A typical scenario generated from Gauss-Markov model is shown in Figure 14.

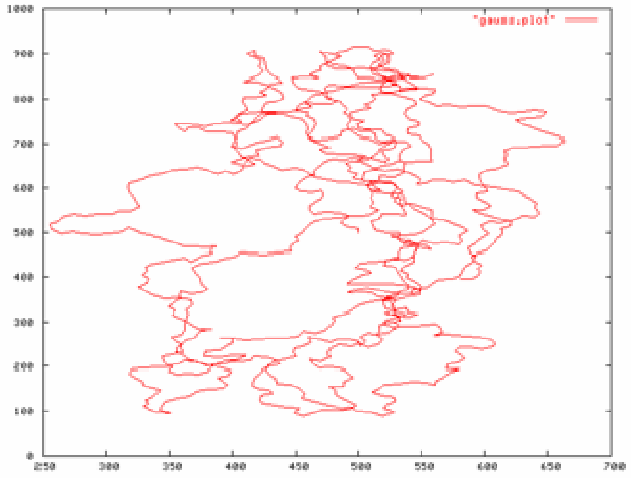


Figure 14 – Gauss-Markov mobility model scenario (from ref. [25])

The implementation of the original Gauss-Markov model follows the publication of B. Liang and Z. Haas [24]. This model assumes that node movement has a pre-determined destination. Within a short period of time, the change in velocity will be limited due to physical constraint. Hence, the next location (and velocity) of a particular node is likely to be correlated to the previous and current position. The parameters required are a norm and random vector which are assigned to each mobile node, instead of stating the velocity. A maximum speed can also be specified. During simulation run, a velocity vector with a larger norm is multiplied with an appropriate scalar quantity to reduce the speed. When a node is the approaching boundaries and about to travel beyond the boundaries, the direction of movement is forced to flip by 180 degree (i.e. mirrored). This keeps the nodes in the boundary of the simulation area.

The Gauss-Markov model is similar to the description from the T. Camp, J Boleng and V. Davies publication [23]. There are subtle differences in the implementation of this model. During simulation, the new speed and movement direction is selected from a normal distribution of previous values, rather than depending on the velocity at a previous instance and a random variable. The speed can also be constrained within a specified range. When the speed exceeds this range, the minimum or the maximum value is selected. As with the original Gauss-Markov model, nodes can move beyond the simulation boundaries and the next movement vector is updated with an angle that brings the node back into the simulation area.

c. Manhattan Grid Model

The Manhattan Grid model is a model where the mobility of nodes is confined in a pre-defined grid area within a topological infrastructure [26]. This model specifies that the movement of nodes is only allowed along paths of pre-defined grid (i.e. parallel movement in the x or y direction). Figure 15 shows the scenario from a Manhattan Grid model. The parameters that are used to describe the model include the mean speed, minimum speed (with a defined standard deviation for speed - normal distribution), probability of speed change at position update and probability that node will turn at cross junctions.

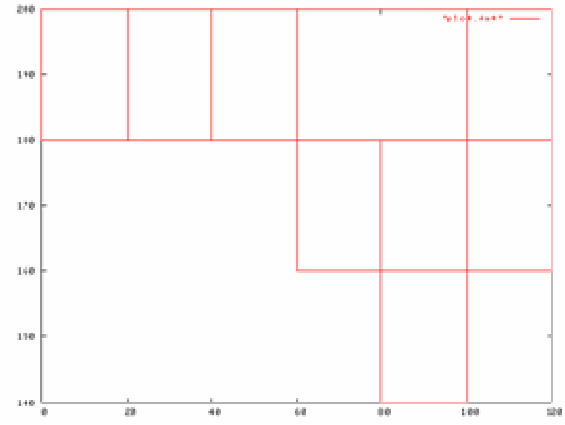


Figure 15 – Manhattan Grid model scenario (from ref. [25])

Such a model is useful in emulating the movement pattern in an urban setup where grids are composed of a number of horizontal and vertical streets, intersecting each other. Node movement is somewhat driven by a destination and physical constraints within the environment. When a mobile node is at an intersection of a horizontal and a vertical street, it can turn left, right or go straight depending on a certain probability.

d. Reference Point Group Mobility Model

Reference Point Group Mobility (RPGM) model [27], unlike previous models, is a group mobility model and allows independent random motion behavior for each node, in addition to the group motion (see Figure 16). Nodes are formed into groups where each group will have a logical centre. The group movements will be based upon the path traveled by a logical center. This model represents the randomness in movement of groups as well as individual nodes within the group. Group movement is determined by the group's logical centre and the motion of nodes in a group is characterized by the centre direction and speed. Within a group, individual nodes move depending on a pre-defined reference point which follows the group movement.

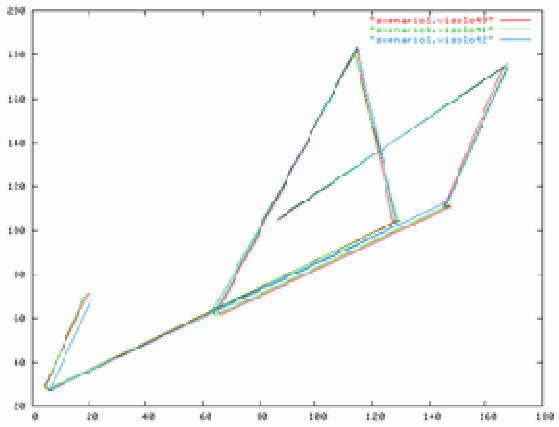


Figure 16 – RPGM model scenario (from ref. [25])

This model is parameterized by the number of nodes per group, group change probability (where nodes *migrate* from one group to another), maximum distance to center of group and group size standard deviation. The RPGM model provides a general and flexible framework for describing mobility patterns that are task oriented and time restricted. Such a model is suitable for military operation environment where military forces usually move in groups.

3. Summary

Although the RWM model is simple and widely used, it does not capture the essence of node movement on the ocean surface. The ocean current is determined mainly by the wind and not necessarily random in movement pattern. Both the Gauss-Markov models constrain the movement of nodes to be within a certain operating environment. Such properties are ideal for simulating a wireless personal communication service that is restricted by wireless coverage. However in oceanic environment, nodes could be washed ashore or drift further into the ocean. The Manhattan mobility model has the properties of high spatial and temporal dependency with a certain restriction on mobility defined by the grid. For modeling of ocean surface current, the movement of nodes is not constrained in any particular fashion except at the coast line. The RPMG is the most promising among the previous models as the nodes in an ocean environment could possibly form group since the difference in velocity of adjacent nodes is small. However, the velocity change is not random but deterministic and is influenced by surface current depending on the time of day. Hence, the models in BonnMotion do not accurately represent the ocean environment and cannot be used to provide meaningful evaluation of IEEE 802.15.4 protocol.

III. NS2 SIMULATION ENVIRONMENT

In computer network research, the uses of network simulation techniques allow simulating network behavior by mathematical calculation of network interaction. Network simulator (NS2) is such an open source simulation tool that operates on the UNIX-based operating systems. This chapter describes NS2, the methodology used to compute node movements, generation of NS2 movement and traffic pattern scripts as well as the performance matrices for evaluation of IEEE 802.15.4.

A. NETWORK SIMULATOR

The network simulator, NS2, is a discrete event simulation tool for network simulation which will be used for evaluating the performance of IEEE 802.15.4. The NS2 software and documentation is currently maintained by University of Southern California's Information Sciences Institute (USC/ISI). NS2 began as a variant of REAL [28] which is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks. It is open-source and operates on the UNIX-based operating systems providing substantial support for simulation of routing, multicast protocols and IP protocols over wired, wireless and satellite networks. In addition, it has the capability to generate graphical details of network traffic (through the Network Animator, NAM) and supports several routing and queuing algorithms. It is also possible to run NS2 on Windows machine using Cygwin application, which provides a Linux-like environment under Windows, or simply by operating NS2 through a VMware player. Installation of NS2 can be done by downloading the all-in-one version or downloading individual components and then compiling each component in a single directory. Despite its capabilities, NS2 has been built by various developers and contains several inherent known as well as unknown bugs.

1. NS2 Architecture

NS2 is written in the C++ programming language with the Object Tool Common Language (OTCL) as the front-end interpreter. A class of hierarchy supported in C++ is the compiled hierarchy and the interpreter hierarchy for OTCL. There will be objects that are completely implemented in C++ or OTCL while some can be implemented in both languages. For objects that are implemented in both languages, there is a corresponding link between the two hierarchies. Figure 17 shows the architecture of NS2.

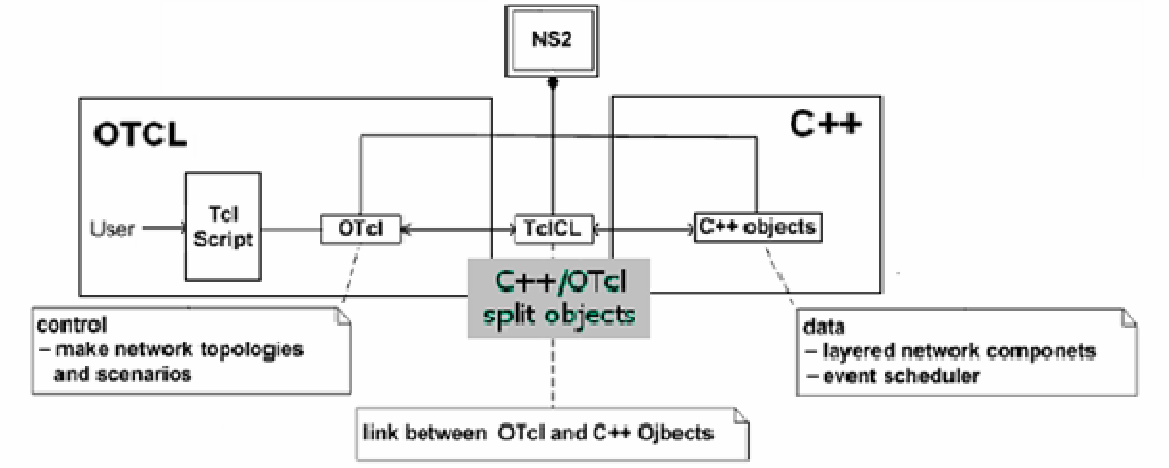


Figure 17 – Architecture of NS2

Basically, the two languages are required to complement both run-time speed and iteration time. A detailed simulation involving efficient manipulation of bytes, packet headers and implementing algorithms with large data set will require run-time speed. The importance of these tasks requires a system programming language similar to C++ programming. On the contrary, where certain parameters might need to be varied for research, iteration time is vital. A scripting language like OTCL is ideal for such a task that allows frequent configuration changes. A scheduler maintained event timing that determines the advancement of the simulation, as shown in Figure 18. An event is an object in the C++ hierarchy with a unique identity, a scheduled time and a pointer to an

object that handles events. For every event created by the network component, the event will be placed onto an event queue linked to the scheduler. The data structure is ordered to be synchronized with the executing events invoked by the event handler. As the simulation progresses, the first event in the queue is assigned with the appropriate network component and executed.

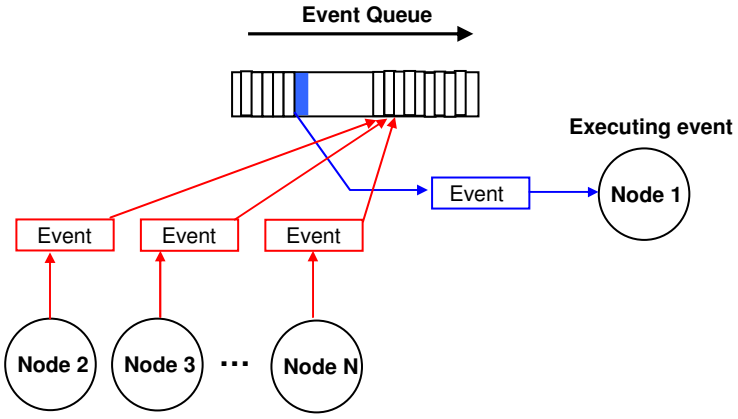


Figure 18 – NS2 Event Scheduler (from ref. [29])

2. Simulation Environment

The version in use for this thesis is version 2.29, with installation of the all-in-one package that operates on Redhat 9.0 through VMware server. The installation process is available in Appendix A. Basically, the scenarios are implemented with scripts written in TCL that comprise commands and parameters for simulator initialization, node creation and configuration. There are three inputs to the simulator; the movement pattern file, the communication pattern file and the configuration file. The movement pattern file describes all node movements while the communication pattern file describes the packet workload presented at the network layer during simulation. These two files essentially constitute the description of the simulation scenario. The final input is the configuration file that defines the ad hoc network routing protocol which is often the main file where the scenario files are called. The results from the simulation are generated in two separate files, an output trace file (*.tr) and a

NAM trace file (*.nam). The trace file will contain information on the various events which occurred, details of node behavior, packet transmissions and receptions, communication layer, packet drops, etc. Analysis of these packets can determine the performance effects from parameter variation, routing protocols and are performed using awk command, perl scripts or available analysis programs. The nam file contains information on the topology such as node movement traces and events. It is similar to trace files with the exception of the syntax that is used by the network animator (NAM) for visual representation of the simulated scenario. The procedure for running the scenarios is shown in Figure 19.

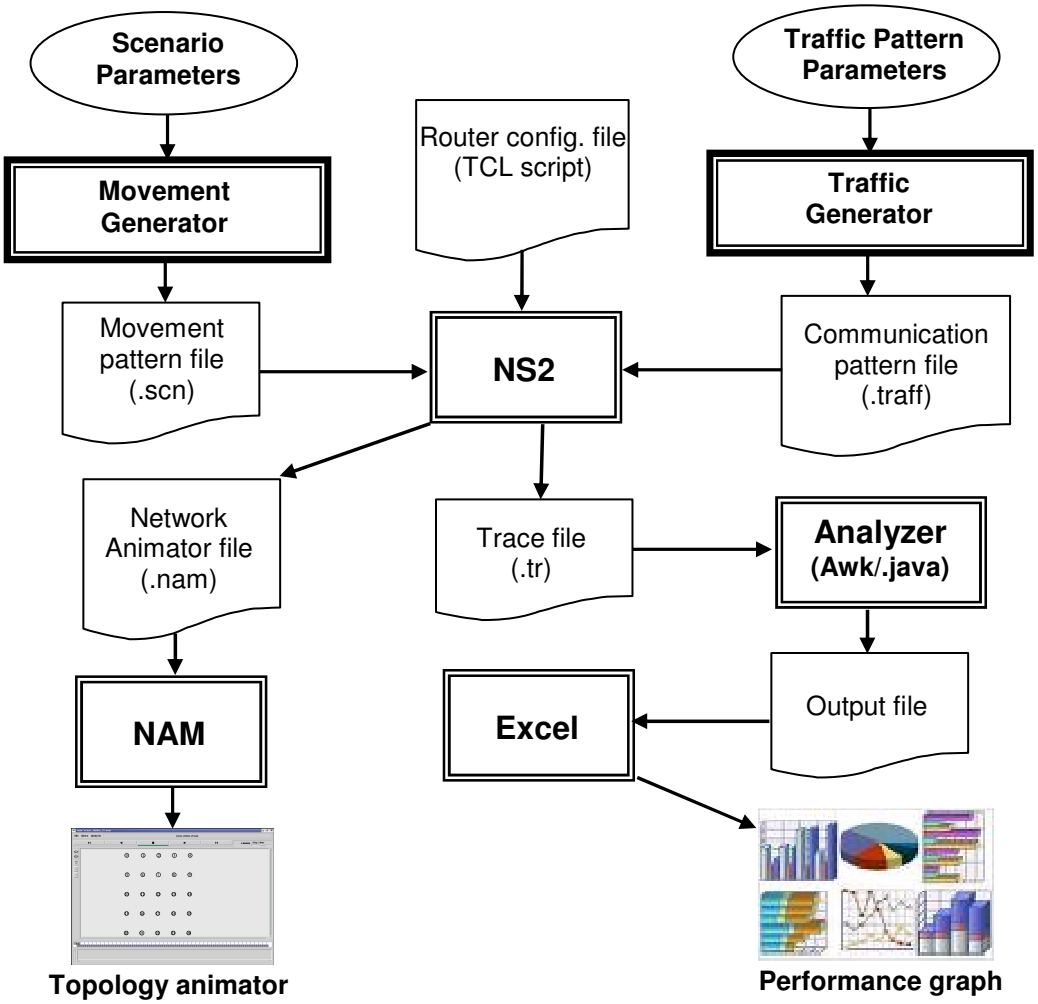


Figure 19 – Flow Diagram for Running Scenario in NS2

3. NS2 Mobile Node Model

The mobile nodes model is part of CMU's mobility extension [30] to NS2 where each mobile node operates independently to compute its position and velocity as a function of time. Mobile nodes are attached to a channel through network interfaces that carry packets between nodes. Figure 20 shows the mobile node. As part of the mobility extension, added functionalities like node movement, transmission and reception on a channel are included to create a wireless mobile environment.

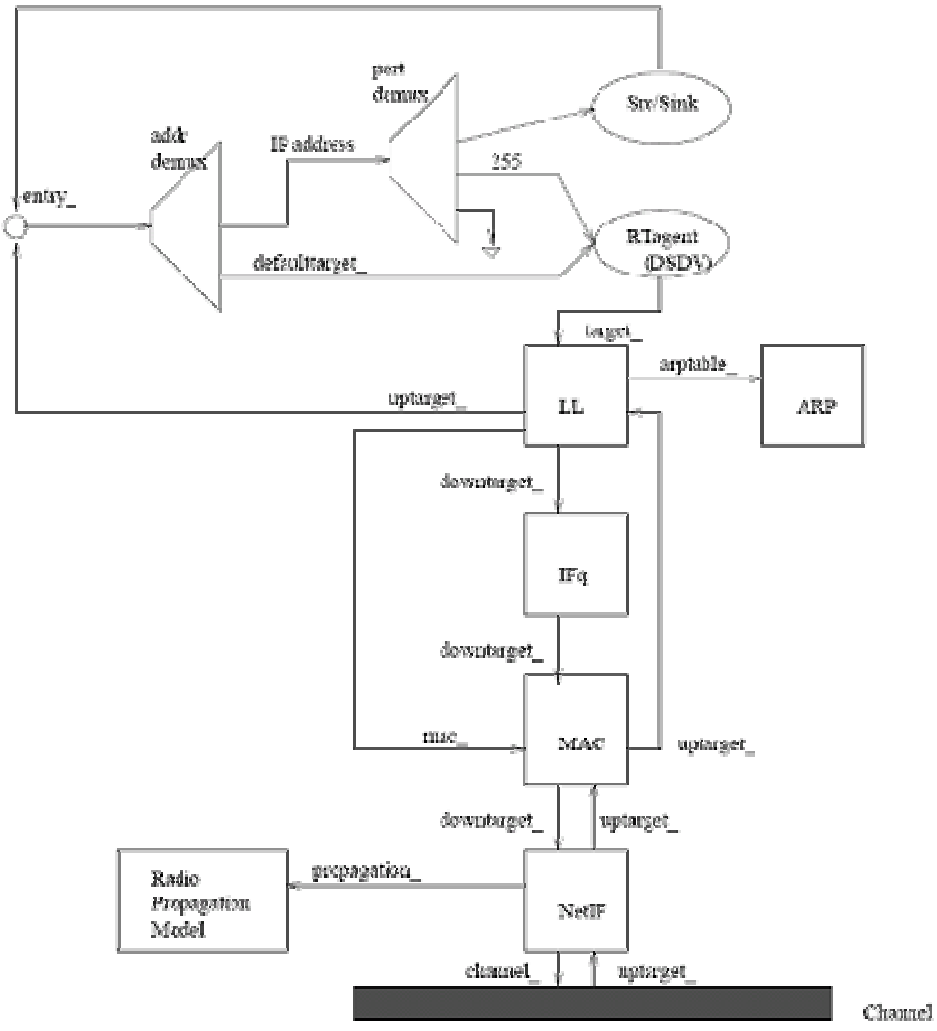


Figure 20 – NS2 Mobile Node Model (from ref. [30])

Features including node movement, periodic position updates and maintaining topology boundary are implemented in the C++ portion whereas the network components (such as classifiers, LL and MAC) are implemented in OTCL. Applications such as constant bit rate (CBR) packets are bounded to a particular node and a routing agent. Routing agents are used by the nodes to compute the routes from source to destination. The packet is then passed to the link layer (LL) and queued until a signal is received from the MAC layer for transmission on the channel. During transmission, copies of the packet are distributed by the channel to all interfaces residing on the channel. Whether a particular interface receives the packet is determined by each network interface's information and radio propagation model that simulate the physical nature of wireless communication. The simulation parameters for this thesis are described in Table 2. Identifying the correct parameters for the simulation is essential for realistic analysis of the study. There are two MAC layer protocols implemented for mobile networks, 802.11 and TDMA. However, the simulation for this thesis is based on IEEE 802.15.4 protocol which is part of the NS2 contributed code.

Traffic parameters	
Traffic type	Constant Bit Rate (CBR) application based on simple User Datagram Protocol (UDP). Using CBR allows the network to be tested for congestion that will affect delivery ratio.
Active flow	Active flow is the communication that is established between a node and the coordinator. Fifteen active flows are used for our simulation.
Packet size	This is the payload of a packet that excludes the MAC and PHY layer headers.
Traffic direction	All the traffic flow from various nodes to the PAN coordinator.
Node parameters	
No. of nodes	There are total of 25 nodes and one of them being the PAN coordinator.
Node movement	Two types of scenario will be simulated, stationary nodes and node movements in ocean environment.
Node position	The nodes are placed 15m apart form each other. The PAN coordinator is the node at the center.
Physical parameters	
Radio Propagation Model	There are three models supported, Free-space, Two-ray ground and shadowing model. Given that nodes may not have direct line-of sight and communication range is limited, the radio propagation model for our simulation is the Two-ray ground model approximation assuming reflection off a flat plane.

Antenna	The antenna used is omni-directional having unity transmitter and receiver gain.
Path loss	The path loss is defines as the attenuation during transmission due to factors like antenna height, obstruction and etc. It is assumed that there is no attenuation and hence the path loss is 1.
Routing layer parameters	
Queue type	DropTail is the queue type employed that implements the "First in First out" techniques. Packets entering the queue will be executed first and once the queue limit is reached, last packet entering the queue will be dropped.
Queue length	The queue length of 100 packets is used as the default parameter. With a WPAN, a large queue length is not required as the data rate is low.

Table 2. Simulation Parameters

4. Setting User Parameters in NS2

For any simulation run, there is a set of control parameters to be specified by the user as well as output parameters that need to be analyzed. As mentioned earlier, the scenario which defines the environment is included in the movement pattern file and communication pattern file (detailed in the following section). The configuration file is essentially a script that consists of a general format that can be used to simulate any kind of routing protocol. The script begins with the definition of the wireless physical medium parameters and initialization of simulation parameters. After setting up the initial parameters, the simulator objects will be set up where the simulator instance, trace object for the network simulator and network animator as well as the General Operations Director (GOD) will be created. GOD is a simulator object, which is used to store global information about the state of the environment, network, or nodes. The topology is also defined where a load_flatgrid object specifies a 2-D terrain. Next, the properties of individual mobile nodes in the ad hoc network are defined as shown in table 2. Each node is then attached to the channel with movement and traffic specified in the movement pattern file and communication pattern file respectively. Finally, information to end the simulation is included. A sample of the script used for the simulation is attached in Appendix D.

5. Post Analysis

At the end of the simulation, the two generated files can be used for performance analysis. The Network Animator file (*.nam) is utilized by the Network Animator (NAM) which is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces [31]. It is created to allow accommodation of large animation data sets and adaptable to different network visualization situations. Simple animation event commands are read from nam files to minimize memory usage. NAM supports topology layout, packet level animation and various data inspection tools. Upon startup, NAM will read the nam file and create the topology followed by displaying the layout on a window. Through the user interface, NAM provides several functions to control the animated simulation. A screenshot of NAM is shown in Figure 21.

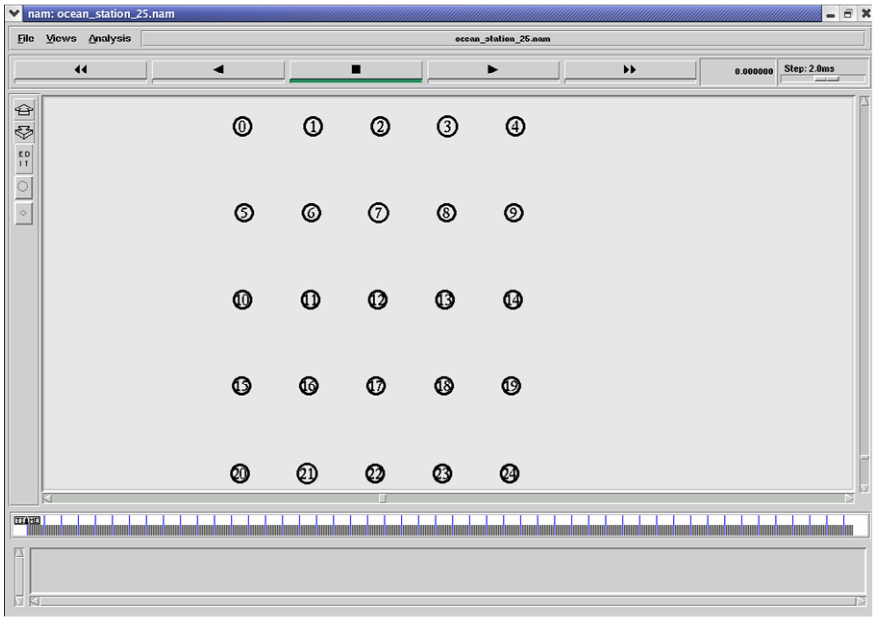


Figure 21 – NAM Application Window

There are two types of trace formats that can be used in NS2. The trace files obtained from the simulation contains the packet dumps between nodes communication. The old format for wireless simulation traces begins with either

one of the characters s, r, D or f to denote sending, receiving, dropping and forwarding (i.e. node not originator) a packet respectively. The old trace format is as shown in Table 3.

Event	Abbreviation	Type	Value
Wireless Event	Commence with : s: Send r: Receive D: Drop f: Forward	%.9f %d (%6.2f %6.2f) %3s %4s %d %s %d [%x %x %x %x]	
		%.9f _%d_ %3s %4s %d %s %d [%x %x %x %x]	
		double	Time
		int	Node ID
		double	X Coordinate (If Logging Position)
		double	Y Coordinate (If Logging Position)
		string	Trace Name
		string	Reason
		int	Event Identifier
		string	Packet Type
		int	Packet Size
		hexadecimal	Time To Send Data
		hexadecimal	Destination MAC Address
		hexadecimal	Source MAC Address
		hexadecimal	Type (ARP, IP)

Table 3. NS2 Trace Format for Wireless Packet

As a part of the effort to merge wireless traces, a new trace format [32] has been introduced. This revised format supports backward compatibility with previous trace formats and is divided into the following fields. The type field is used to differentiate among different types of traces. As with the old format, the new format has similar field types. Typically, the field type is followed by general flag (-t) and sets of flag/value pairs with the first alphabet of the flag type representing node property (N), IP level packet information (I), next hop information (H), MAC level packet information (M) and packet specific information (P) shown in Table 4.

Event	Abbreviation	Flag	Type	Value
Wireless Event	s: Send r: Receive d: Drop f: Forward	-t	double	Time (* For Global Setting)
		Next hop information		
		-Hs	int	Hop source node ID
		-Hd	int	Hop destination Node ID -1: the packet is a broadcast packet -2 destination node has not been set
		Node properties		
		-Ni	int	Node ID
		-Nx	double	Node X Coordinate
		-Ny	double	Node Y Coordinate
		-Nz	double	Node Z Coordinate
		-Ne	double	Node Energy Level
		-Nl	string	Network trace Level (AGT, RTR, MAC, etc.)
		-Nw	string	Drop Reason
		Packet information at MAC level		
		-Ma	hexadecimal	Duration
		-Ms	hexadecimal	Source Ethernet Address
		-Md	hexadecimal	Destination Ethernet Address
		-Mt	hexadecimal	Ethernet Type
		IP Trace		
		-Is	int.int	Source Address And Port
		-Id	int.int	Destination Address And Port
		-It	string	Packet Type (cbr, tcp, …)
		-Il	int	Packet Size
		-If	int	Flow ID
		-Ii	int	Unique ID
		-Iv	int	TTL Value
		Packet info at Application level		
		-P	string	Packet Type (arp, dsr, imep, tora, etc.)
		-Pn	string	Packet Type (cbr, tcp, …)

Table 4. NS2 New Trace Format for Wireless Packet

Depending on the packet type, the trace may log additional information. For the simulation performed, there are three packet types, ARP, CBR and AODV. The format for the ARP, CBR and AODV trace are shown respectively in the tables below.

Event	Flag	Type	Value
ARP Trace	-Po	string	Request or Reply
	-Pms	int	Source MAC Address
	-Ps	int	Source Address
	-Pmd	int	Destination MAC Address
	-Pd	int	Destination Address

Table 5. ARP Trace Format

Event	Flag	Type	Value
AODV Trace	-Pt	hexadecimal	Type
	-Ph	int	Hop Count
	-Pb	int	Broadcast ID
	-Pd	int	Destination
	-Pds	int	Destination Sequence Number
	-Ps	int	Source
	-Pss	int	Source Sequence Number
	-Pl	double	Lifetime
	-Pc	string	Operation (REQUEST, REPLY, ERROR, HELLO)

Table 6. AODV Trace Format

Event	Flag	Type	Value
CBR Trace	-Pi	int	Sequence Number
	-Pf	int	Number Of Times Packet Forwarded
	-Po	int	Optimal Number Of Forwards

Table 7. CBR Trace Format

For analyzing the performance of IEEE 802.15.4, information has to be extracted to compute the performance parameters from trace files using AWK or Perl script. AWK is a programming language designed for processing text-based data, either in files or data streams. Perl is a dynamic programming language that complements AWK to allow processing of repetitive simulation runs. Both

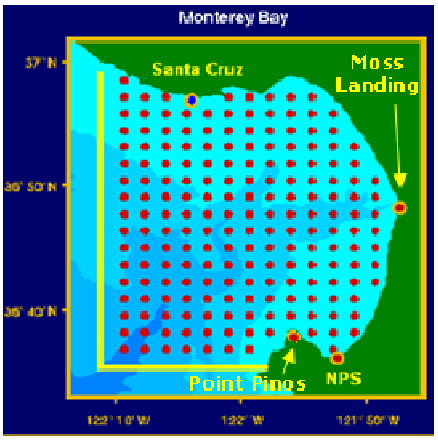
languages are available in standard UNIX environment. Java program can also be implemented to analyze the trace files.

B. GENERATING OCEAN ENVIRONMENT SCENARIOS

As the above mobility models do not correctly represent the oceanic environment, there is a need for other means to generate the movement pattern file. Another alternative is to compute node movements using actual surface current measurements. In NS2, movement files are generated from mobility models or by using node movement generator. The node-movement generator allows configuring of nodes movement at particular time and is available under ~ns/indep-utils/cmu-scen-gen/setdest directory consisting of setdest{.cc,.h} and Makefile. The setdest function can then be used to create the movement scenario for NS2. The other part of the scenario is to define and generate the communication pattern file.

1. Near Real-Time Monterey Bay Surface Currents

Near real-time coastal surface current data of the Monterey Bay (California) are made available from a collaboration between the Naval Postgraduate School, University of Delaware and Monterey Bay Aquarium Research Institute [33]. Four High Frequency (HF) radars at different locations measure the currents and the measurements are objectively mapped to fill gaps in space and time. Mapping is required due to environment factors causing unavailability and spatial coverage of the HF radar measurements. The results of hourly mapped total vectors will be used for the computation of node movements. The ASCII data files contains the hourly data of pre-determined locations in latitude/longitude (lat/long) with the vector speed (cm/s) shown in Figure 22.



Longitude	Latitude	U (cm/s)	V (cm/s)
-122.1869	36.5899	23.8248	-7.5657
-122.1807	36.5899	30.0602	-15.5001
-122.1744	36.5899	29.2442	-15.9289
-122.1682	36.5899	28.6670	-16.3433
-122.1620	36.5899	28.2209	-16.7318
-122.1557	36.5899	27.8382	-17.0822
-122.1495	36.5899	27.4390	-17.3813
-122.1432	36.5899	26.9970	-17.6144
-122.1370	36.5899	26.4484	-17.7648
-122.1308	36.5899	25.7658	-17.8137
-122.1245	36.5899	24.9905	-17.7392

Figure 22 – Data points of Monterey Bay

2. Determining Mobility Using Actual Measurements

A sample data set (dated 30 January 2005) of the Monterey bay area was taken to simulate the nodes' movement. Since the variation is small across each time step, the hourly velocities are linearly interpolated for the entire day at a time interval of 10 minutes. Computation was performed for an entire day at a time interval of ten minutes. Basically, the velocity vectors of the data points nearest to the node were referenced to provide a random velocity from a normal (Gaussian) distribution. The randomness was included to account for turbulent diffusion and was applied to the node. This turbulent motion included the meandering of the current, tides, waves, and random motion which caused the nodes to disperse. The next position of the node was computed and the computation was repeated until the end of the day. The following sub-paragraphs detail the steps (performed using Microsoft Excel) to compute a node movement when placed within the Monterey Bay area.

Step 1: The initial position (lat/long) of the node is searched through the data files for nearest data points beginning at 0000hrs.

Step 2: The velocity of the data points nearest to the node were referenced and a random velocity was generated from a normal (Gaussian) distribution which was applied to the node. This velocity then computed the distance and direction of the next position for a period of 10 minutes.

Step 3: The coordinates of the next position were computed using the formula [34] given below. The computation repeated until the boundaries of the bay were exceeded or simulation time (one day) was reached.

$$Latitude\ (x) = \arcsin\left(\sin\ x'\cos d + \cos x'\sin d\cos t_c\right) \quad (1)$$

$$Longitude\ (y) = \mathrm{mod}\left(y' - \arcsin\left(\sin t_c\ \sin d / \cos x\right) + \pi,\ 2\pi\right) - \pi \quad (2)$$

where

x' *is the latitude of the current location*,
y' *is the longitude of the current location*,
d *is the distance in radians*,
t_c *is the true course heading in radians*.

The result of simulating 25 nodes placed 10m apart from adjacent nodes within the Monterey Bay is shown in Figure 23. Initially, nodes can be seen to be moving towards the shore and around 1100hrs, nodes start to drift out towards the ocean. Nodes are gradually dispersing away from each other until the end of the day. Such a dispersing behavior is consistent with the meandering of the current, tides, waves, and random motion. Also, since the nodes are within the bay area, the movements are very much dominated by tidal currents. This pattern is unique which cannot be correctly modeled by any models. Hence, the data (node position at various times and speed) obtained from the computation will be used for the simulation.

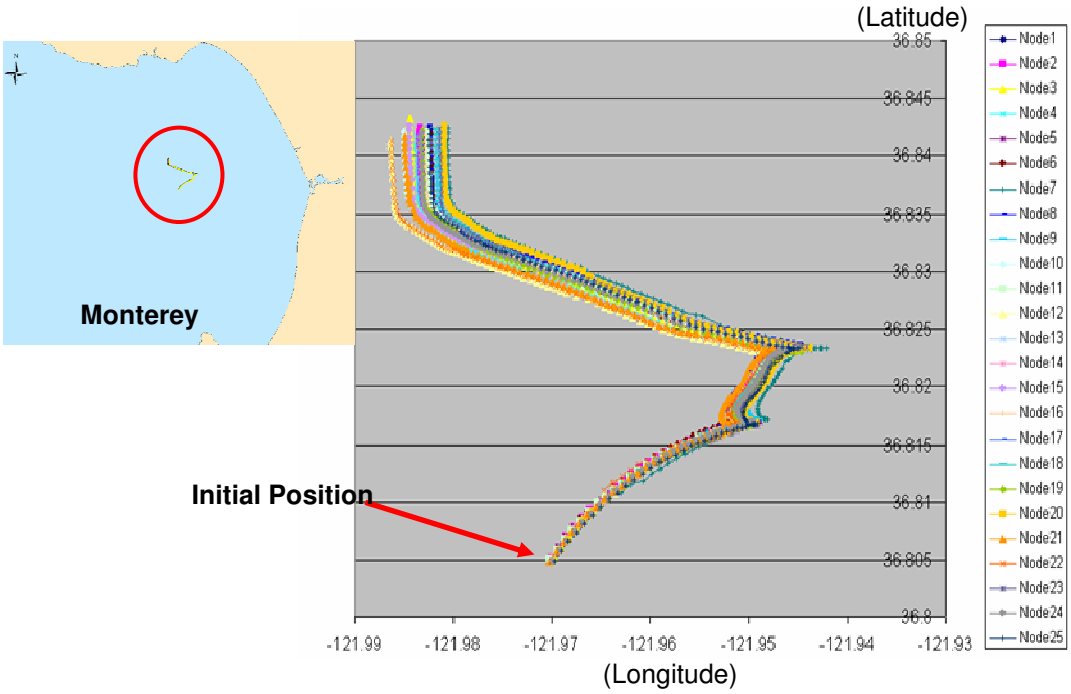


Figure 23 – Node movements in Monterey Bay

3. Generating Movement Pattern File

A movement pattern file is required as an input to NS2 for the simulation. An external script (see Appendix B) is implemented to parse the computed movement data and generate the mobility file. The script is written in Tool Command Language (TCL) and essentially functions to format the data into the movement file. The movement file consists of initial position of each node (two dimensional) and the “setdest” commands for each movement change at stipulated time. Subsequently, the formatted file can be used to calculate information for GOD (General Operations Director) object which can also be present in the movement. The “calcdest" function (found in ~ns/indep-utils/cmu-scen-gen/setdest/ directory) is used in which information of the GOD object is computed and loaded into the movement file. The GOD object is initially used to store global information about the state of the environment, network, or nodes

but currently stores an array of the shortest number of hops required to reach from one node to another. The calculation is performed in advance to minimize computation time required during simulation.

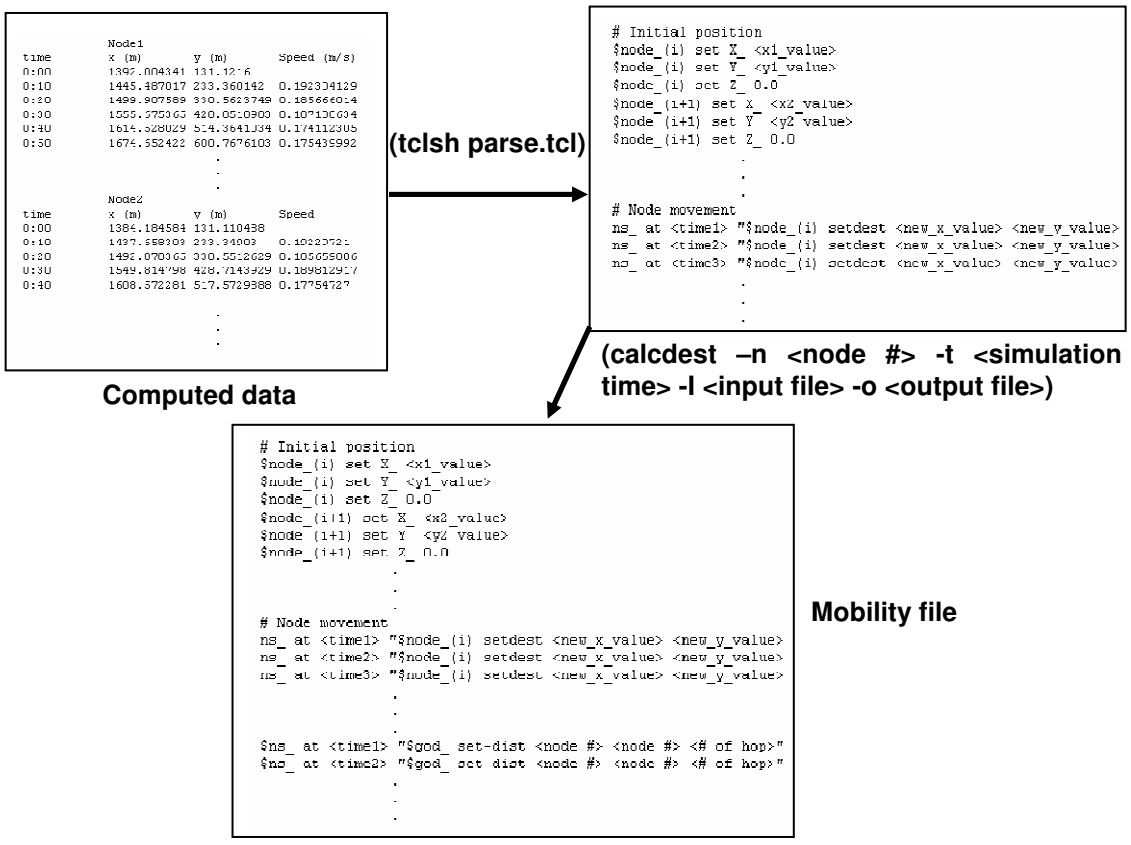


Figure 24 – Process of Generating Mobility File

Figure 24 depicts the process and results generated from the script. It was discovered that the nodes had to be arranged in an anti-chronological order for the correct execution of “calcdest” function. The script below is modified from a sample script provided by Matthias Budde in the NS forum, customized to our computed data. This script essentially parses the numerous data generated by the node movement computation using the “Tclsh” command. “Tclsh” is a shell like application that reads and evaluates TCL commands from standard input or a file. Data points are extracted and arranged in the required format of the

movement file. The computed movement data have to be arranged in anti-chronological order for subsequent computation of GOD object.

4. Generating Communication Pattern File

The communication pattern file essentially describes the packet workload presented at the network layer during simulation. A TCL script is modified from a constant bit rate generation script file [35] specifically design for star topology. The modification includes change to generate traffic at an interval of 10 minutes which coincide with node movements where nodes will be randomly selected to transmit data to the PAN coordinator (see Appendix C). The script can be executed using “ns” command, followed by optional parameters. These parameters specify the traffic type (cbr or tcp), number of nodes in the scenario, seed for generating random traffic pattern, maximum traffic flow, traffic rate, start time, end time and the time gap between consecutive transmissions. The traffic scenario is then piped to a communication pattern file.

C. PERFORMANCE METRICS

In this thesis, the performance of IEEE 802.15.4, in particular, the Quality of Service (QoS) is to be evaluated against the oceanic environment. The QoS is determined by a set of service requirements that the network must fulfill when transporting packet streams from a source to its intended destination. Due to the mobility and dynamic nature of the MANET, resources might not always be available and parameters such as throughput, delay and packet drop have to be considered. Essentially, two scenarios are used for the simulations, stationary nodes and mobile nodes floating on the ocean. Three performance metrics will be used to determine the QoS provided by IEEE 802.15.4. A detailed explanation of these metrics is as follows:

1. Packet Delivery Ratio

Packet Delivery Ratio (PDR) is the quotient resulting from the number of successful delivered CBR packets to those generated by CBR sources within the simulation period. It is an important metric which indicates congestion level of the network. PDR measures the protocol performance from loss ratio experienced at the network layer that is affected by factors such as packet size, network load and effects of movement resulting in frequent topological changes. Higher PDR implies that packet loss rate is lower and protocol is more efficient from the perspective of data delivery. A point to note is that late packet received could be deemed useless even with high PDR.

$$PacketDeliveryRatio\ (PDR) = \frac{\sum Number\ of\ \ received\ \ CBR\ \ packets}{\sum Number\ of\ \ transmitted\ CBR\ \ packets} \tag{3}$$

2. Average Network Delay

Network delay is the time delay experienced by a connection between nodes. This delay includes all possible delays that are caused by route discovery latency, queuing in the interface queue, retransmission at the MAC layer and propagation through the environment. This delay can be computed as the difference in time from transmitting packets from a source to the arrival of packets at the destination node. All the connection delays are aggregated and the average network delay is the aggregated delay divided by the number of connection pairs throughout the simulation.

$$Average\ Network\ \ Delay = \frac{\sum \left(Time_{\ packet\ arrive\ @\ dest} - Time_{\ packet\ sent\ @\ source} \right)}{Total\ number\ of\ \ connection\ \ pairs} \tag{4}$$

3. Network Throughput

The network throughput determines the amount of data that is transmitted from a source to a destination node per unit time (bits per second). While ignoring the overheads in the network, only the data of the CBR packets are considered. Node throughput is a measure of the total number of data packets successfully received at the node, having the total number of bits is computed over the simulation runtime. The network throughput is then derived from the average throughput of all nodes involved in the CBR packet transmission.

$$Node\ Throughput = \frac{Total\ bits\ received\ by\ node}{Simulation\ time} \tag{5}$$

$$Network\ Throughput = \frac{\sum Node\ Throughputs\ of\ CBR\ transmission}{Total\ number\ of\ nodes} \tag{6}$$

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SYSTEM SIMULATION AND RESULTS

This chapter identifies additional parameters to be used in the simulation with reference to a Zigbee transceiver as well as the analysis of IEEE 802.15.4 against the performance metrics. The performance metrics are used in deriving the performance specific to mobility, node density and network loading. The simulation is reiterated over several runs until certain conclusions can be drawn to determine the feasibility of employing IEEE 802.15.4 technology in the ocean environment.

A. SPECIFIC TRANCEIVER PARAMETERS

Besides physical parameters defined in the previous chapter, the specification of the transceiver plays an important role in determining the communication range of the nodes. The power parameter is based on the specification of Crossbow MICAz processor and radio platform (MPR2400CA).

1. Transmitted Power (Pt_)

The transmitted power is the power that is transmitted from the antenna into space. In Annex F [4], the document states the IEEE 802.15.4 regulatory requirements that for the ISM band 2.4 GHz operating in United States, transmitted power of up to 1 watt is provided. Although IEEE 802.15.4 devices are generally envisioned to operate with a maximum transmit power of approximately 0 dBm (1mW), a minimum of +10 dBm (0.01W) is allowed in this band. The crossbow RF transceiver has power of -24 dBm to 0 dBm, hence 1mW for Pt_ will be used in the simulation.

$$Pt\_ = 0\ dBm = 1\ mW \tag{7}$$

2. Receiver Threshold (RXThresh_)

The receiver threshold is the parameter used to specify the communication range of the wireless nodes and the threshold is the minimal power of the packet required for successful reception. If a packet reaches a node with a power level above the receiver threshold, the receiver will be within the transmission range of the sender. The receiver sensitivity for the crossbow transceiver is -94 dBm typical. Taking the typical receiver sensitivity, the receiver threshold is -94 dBm which is equivalent to 3.981×10^{-13} W.

$$RxThresh\_ = -94dBm = 3.981 \times 10^{-13}\, W \tag{8}$$

3. Carrier Sensing Threshold (CSThresh_)

The carrier sensing threshold describes the sensing range of a node. When the received packets have a power level below the receiver threshold but above the carrier sensing threshold, the receiver is within the carrier sensing range but decoding of the packet is not guaranteed. The sensing range of a node is usually greater than the transmission range. Hence, it is assumed that the carrier sensing threshold value is at least the receiver threshold.

$$CSThresh\_ = RxThresh\_ = 3.981 \times 10^{-13}\, W \tag{9}$$

4. Capture Threshold (CPThresh_)

During packet transmission, it is possible that two packets can simultaneously arrive (i.e. collision) at the receiving node. In NS2, only packets received in the carrier sensing range of a receiver are considered as potential sources of packet collisions. The capture threshold determines if a packet is successfully received by comparing the power ratio of both packets. The packet with the highest ratio that is greater than the capture threshold is chosen. The

capture threshold (CPThreshold) is assumed to be 10 dB in the simulations which is a common value used in most simulations.

$$CPThresh\_ = 10\,dB \tag{10}$$

5. Antenna Height

For the simulation environment, the antenna of a particular node will be mounted on a floating buoy. Assuming that the buoy has a diameter of 1 m and is partially submerged, the antenna will be at a height of approximately 0.5m. In NS2, the antenna is defined as a set of three dimensional coordinates with the z dimensional coordinate being antenna height.

$$\begin{aligned} X\_ &= 0\,m \\ Y\_ &= 0\,m \\ Z\_ &= 0.5\,m \end{aligned} \tag{11}$$

6. Propagation

There are 4 types of models used in NS2 to represent the radio propagation properties of the environment, namely the free space, two-ray ground, Ricean, Rayleigh and shadowing models. The free space model is a large scale model where the received power is dependent on transmitted power, antenna gains and the range between communicating nodes. The model essentially accounts for the decrease in power as the range increases. As with free space model, the two-ray ground model is also a large scale model. The difference being that the two-ray ground model considers the received power from both the direct path and a ground reflection path. However, a limitation posed in NS2 is that both the communicating nodes have to be of the same height. Both the Ricean and Rayleigh are fading models that describe the propagation effect of a RF signal in a defined environment. Rayleigh fading is most applicable when there is no line of sight between communicating nodes

while Ricean fading is more applicable if there is a line of sight. Finally, the shadowing model assumes the average received signal power decreases logarithmically with distance. A Gaussian random variable is added to the path loss to account for environmental influences at the sender and the receiver. Considering the complex ocean environment, the propagation may vary in different location. For simplicity, the two-ray ground model is selected. According to the two-ray ground propagation model, the sensing range of the device is computed to be 112m.

B PERFORMANCE ANALYSIS

In practice, node movements in the ocean will result in the network having to re-learn optimum routes. Several experiments are conducted to analyze the behavioral response of the IEEE 802.15.4 network, utilizing a Zigbee transceiver that is operating at the 2.4 GHz frequency band. Essentially, the influence of mobility, node density and network loading on the network will be studied. In particular, the network's packet delivery ratio, delay and data throughput performance are measured with varying transmission rates. The performances resulting from the metrics are presented with moving scenarios. To increase the confidence level of the results, a set of simulation parameters are performed with various random seeds for the data transmission. However, running the scenarios required long simulation hours for node movements within a day.

1. Simulation Setup and Parameters

Wireless networks based on IEEE 802.15.4 are designed for low data rate and low power consumption devices. Depending on the operating requirements and environment, there has to be compromises between the node transmission range, operational lifespan and device cost. The transmission range can be determined from Friss two-ray ground reflection models which relates the

maximum range to the antenna gain, transmit power and receiver sensitivity. With the Crossbow MICAz processor and radio platform, the maximum range is computed to be approximately 112m. Since the nodes are highly mobile, this range will determine if a data packet can successfully arrive at the destination node. The data traffic type is Constant Bit Rate (CBR) with the application agent sending at a rate of 20 data packets per second continuously. The nodes used in the simulation will be placed at predetermined distances from adjacent nodes covering an area of 40m by 40m. The centre node is designated the PAN coordinator with all other nodes randomly transmitting data packets to the PAN coordinator. The speed and heading of each node will vary according to the generated ocean movement with a maximum simulation runs of 12 hours. The simulations are conducted with the typical parameters shown in Table 8.

Simulation Parameters	
Protocol	AODV
Simulation Time	1hr, 3hrs, 6hrs, 9hrs, 12hrs
# of Nodes	9, 16, 25
Map Size	4500m $\times$ 4500m
Speed	Varying from 0.006m/s-0.47m/s
Mobility Model	Actual Measurement Model
Traffic Type	Constant Bit Rate (CBR)
Packet Size	20 bytes
Connection Rate	0.05 to 5 packets per second

Table 8. Simulation Parameters

During the simulation, certain behaviors are exhibited by the nodes if communication cannot be established with the coordinator. When a node losses synchronization, orphan-scanning is performed to relocate the coordinator. When coordinator relocation is successful, communication will resume. If relocation of the coordinator fails, node will subsequently perform active channel scan to send association request to the PAN coordinator. Upon successful association, the

node begins to transmit beacons and start data transmission. Else, non-beacon mode is enabled for that node.

2. Influence of Node Density

In an ocean environment, the number of nodes providing coverage in an area of operation will determine the performance of sensors (i.e. detection range) used in the nodes and ultimately derive the cost of operation. As such, the performance of the network is measured with varying node densities to ascertain the impact. In this set of simulations, the node density is increased from 9, 16 and 25 nodes within operational coverage of 40m $\times$ 40m. Figure 25 shows the initial node placement within the operating area.

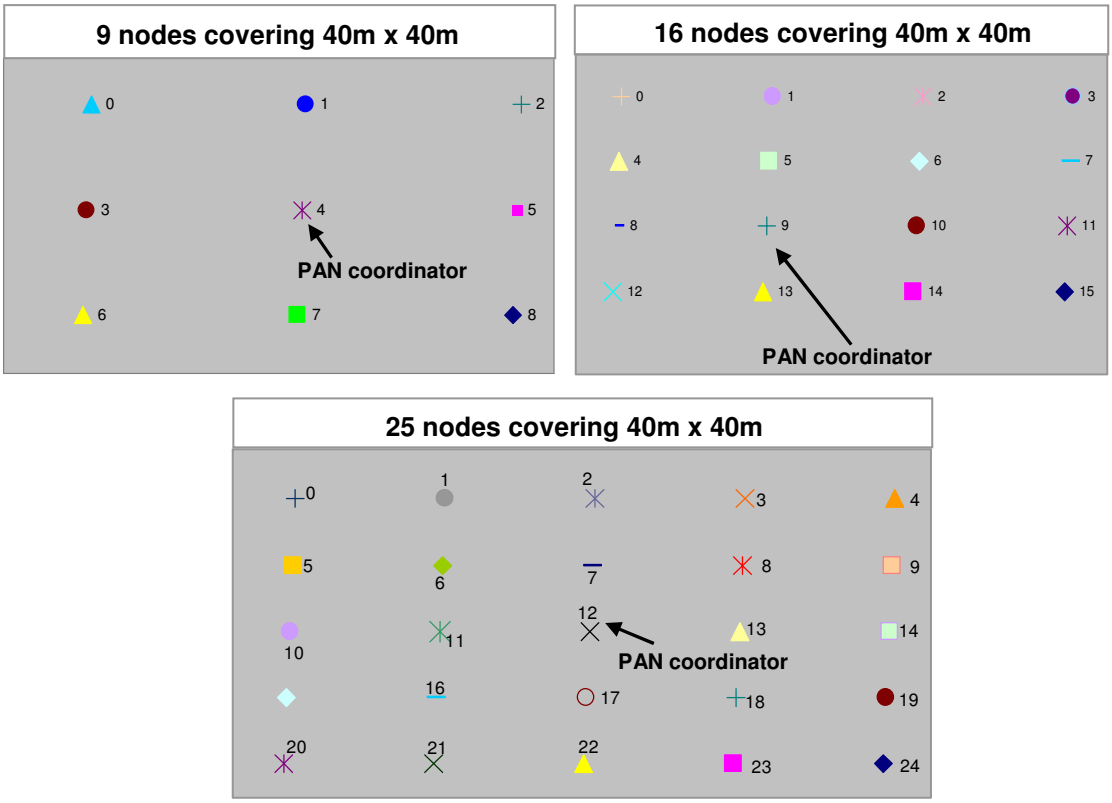


Figure 25 – Initial Coverage with 9, 16 and 25 Nodes

Three sets of scenarios, namely the 1, 3 and 6 hours runtimes, will be used as a basis of comparison with CBR traffic at a data rate of 0.1 packets per second. For each scenario set, the number of connections is set to 8, 14 and 20 respectively where all nodes are transmitting packets to the PAN coordinator. The designated PAN coordinator and nodes positions after the respective simulation runtime are shown in Annex G. The rest of the simulation parameters remain relatively unchanged. Table 9 summarizes the simulation parameters for this set of simulations.

Simulation Parameters	
Protocol	AODV
Simulation Time	1hr, 3hrs, 6hrs
# of Nodes	9, 16, 25
Max Node Connect	8, 14, 20
Map Size	4500m $\times$ 4500m
Speed	Varying from 0.006m/s-0.47m/s
Mobility Model	Actual Measurement Model
Traffic Type	Constant Bit Rate (CBR)
Packet Size	20 bytes
Connection Rate	0.1 packets per second

Table 9. Simulation Parameters for Node Density Analysis

In general, due to the highly dynamic nature of node movement, low density may cause the network to be frequently disconnected resulting in a low throughput. The experimentation goal is to determine the network scalability rather than to find the optimal node density. Figure 26 shows the simulation results which demonstrate the performance metrics using different node densities. An interesting observation is that after an hour of simulation, the packet delivery ratio (PDR) remains close to 100% despite the difference in node densities. This is due to the node transmission range and also the dispersion of nodes since the movements remains insignificant. As the simulation time approaches three hours, there is a slight reduction in PDR but still well within the 90^{th} percentile range.

Thereafter, the PDR falls below 90%.

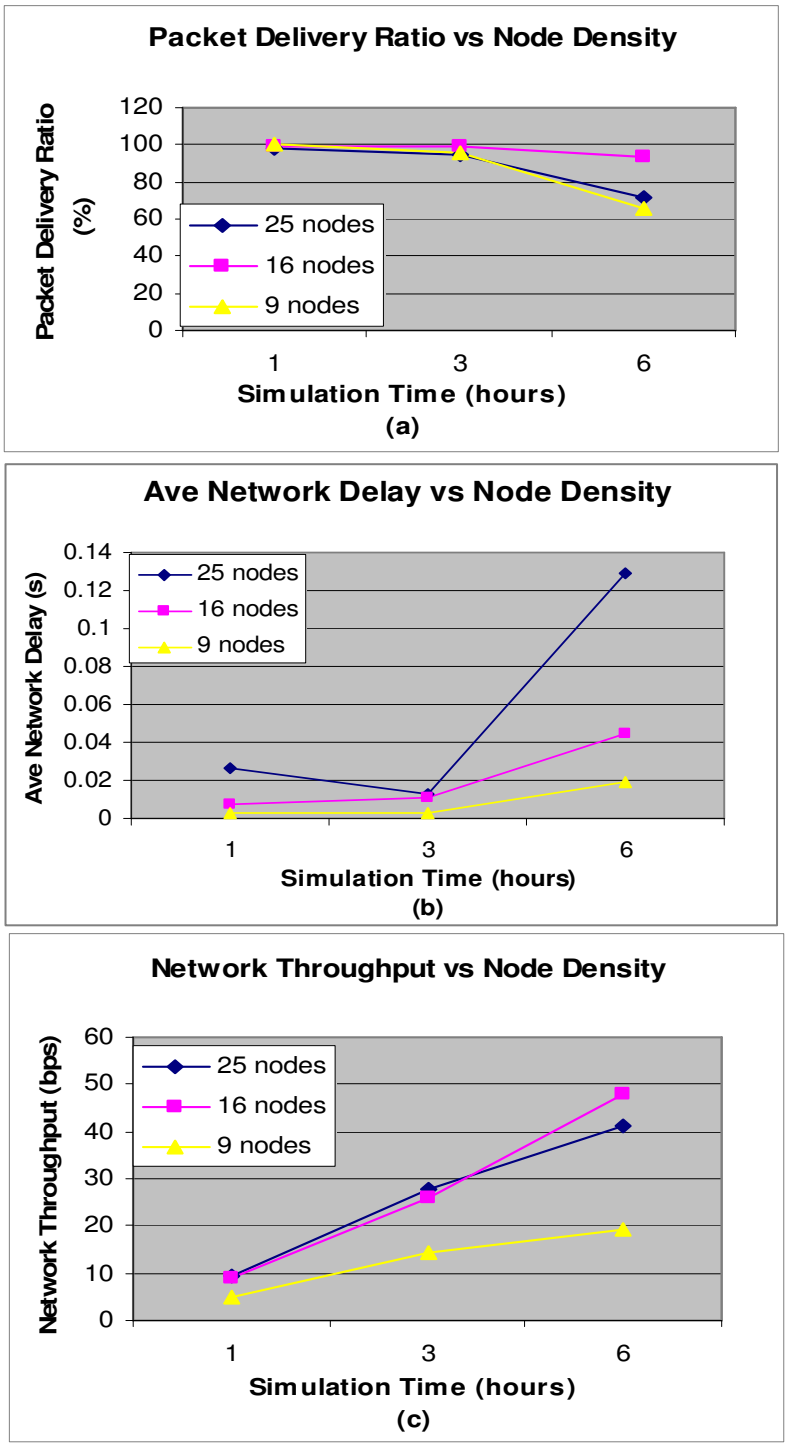


Figure 26 - Results of Varying Node Density with (a) PDR (b) Average network delay and (c) Network Throughput

An interesting observation is that the density of 16 nodes performs better as compared to the others, which is unexpected. This could be due to the maximum number of connections in each scenario. Since all the packets are effectively sent to the PAN coordinator, the 25-node configuration means that more packets are sent to the PAN coordinator resulting in collisions. As for the 9-node configuration, the PDR reduces as the simulation time increases due to the spreading of node movement. The delay measured is the average transmission time of all data packets. The 25-node configuration experiences the most delay since some of the packets are required to be forwarded by adjacent nodes before reaching the PAN coordinator. Also, as the nodes are much dispersed at the three hours simulation time, the effect is multiplied resulting in an increased in average network delay. The throughput effectively determines the amount of data that is transmitted and successfully received during the simulation time. In general, the network throughput increases cumulatively as the time increase. At the 6 hours, the network throughput of the 16-node configuration can be seen to perform better than the 25-node configuration. This observation reinforced the earlier PDR observation since more data is received at the destination. Effectively, the node density directly influences the achievable network throughput. The above findings have indicated the node density does influence the performance of 802.15.4. An ideal node configuration has to be determined considering the operational area as well as the nodes dispersion in the ocean.

3. Influence of Network Loading

In this simulation, the influence of network loading is studied. The emphasis is on the behavior of the network since IEEE 802.15.4 is primarily designed for low bandwidth applications. However, depending on the operational needs, a higher loading might be required in certain situations. The data rate is gradually increased from 0.005 packets per second to 50 packets per second in the three

node configuration scenarios. As in previous simulation, the parameters used remains are relatively unchanged. Table 10 summarizes the simulation parameters for this set of simulations.

Simulation Parameters	
Protocol	AODV
Simulation Time	1hr
# of Nodes	9, 16, 25
Max Node Connect	8, 14, 20
Map Size	4500m × 4500m
Speed	Varying from 0.006m/s-0.47m/s
Mobility Model	Actual Measurement Model
Traffic Type	Constant Bit Rate (CBR)
Packet Size	20 bytes
Connection Rate	0.005 - 50 packets per second

Table 10. Simulation Parameters for Network Loading Analysis

Since all nodes have the PAN coordinator as the destination, the increase in network loading will increase the probability of collision at the PAN coordinator. Figure 27, 28 and 29 depict the effects of network loading on the performance metrics by increasing the connection rate from 0.05 to 50 packets per second. The packet delivery ratio generally has a declining trend for the three node configurations when the connection rate is increased. It can be observed that the packet delivery ratio remain in the 90^{th} percentile range till a specific data rate (approximately 0.3 packets per second, 0.75 packets per second and 1.5 packets per second for the 25, 16 and 9 nodes scenario). Thereafter, the packet delivery ratio performance decreases exponentially for higher data rates indicating that the network is congested. As the data rate increases, more packets will be dropped due to collision and bad link quality of transmitted packets. The resulting effect is the exponential increase in re-transmission of packets and hence further congesting the network. An ideal operating data rate can determine the efficiency of data delivery. For the above reason, regardless of the operational needs, the

application has to maintain a low data rate. In the worst case scenario, the IEEE 802.15.4 protocol cannot be used for data rates above 0.3 packets per second in which performance will be unacceptable.

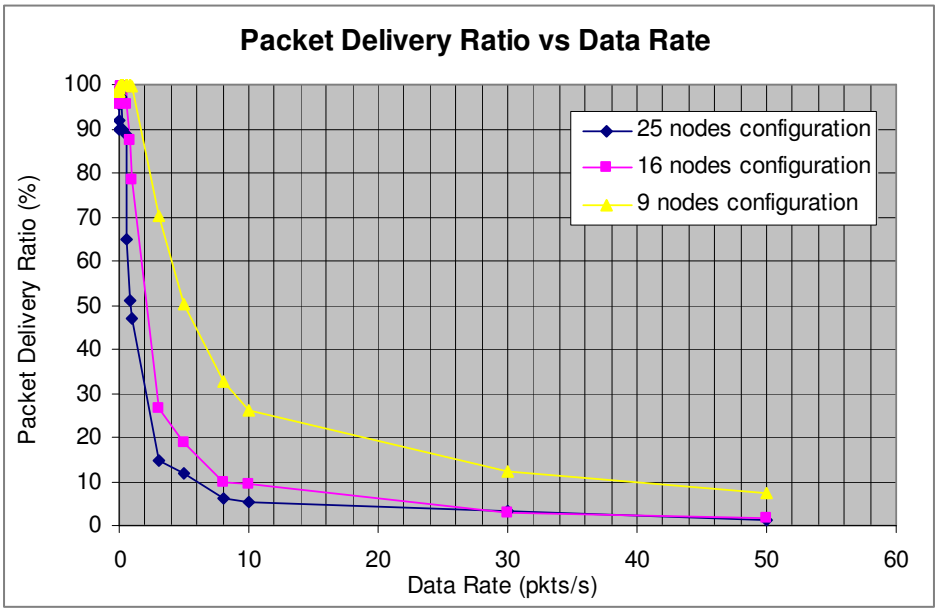


Figure 27 - Results of Packet Delivery Ratio with Varying Data Rate

Theoretically for network delay, the increase in data rate will increase the time for the data packet to reach the intended destination. However, the average network delay from Figure 28 can be seen to be generally low, fluctuating between 0 to 0.6 second. The peaks at low data rate experienced by both the 16-node and 25-node configuration could be due to the time required for packets to be forwarded from outer modes to the destination. The 9 nodes configuration is not affected since packets are sent directly to the PAN coordinator. At higher data rates, collision will occur. However, computation of average network delay does not consider dropped packets hence the delay reduces. Overall, the 25-node configuration experiences a higher delay as compared to the other configurations.

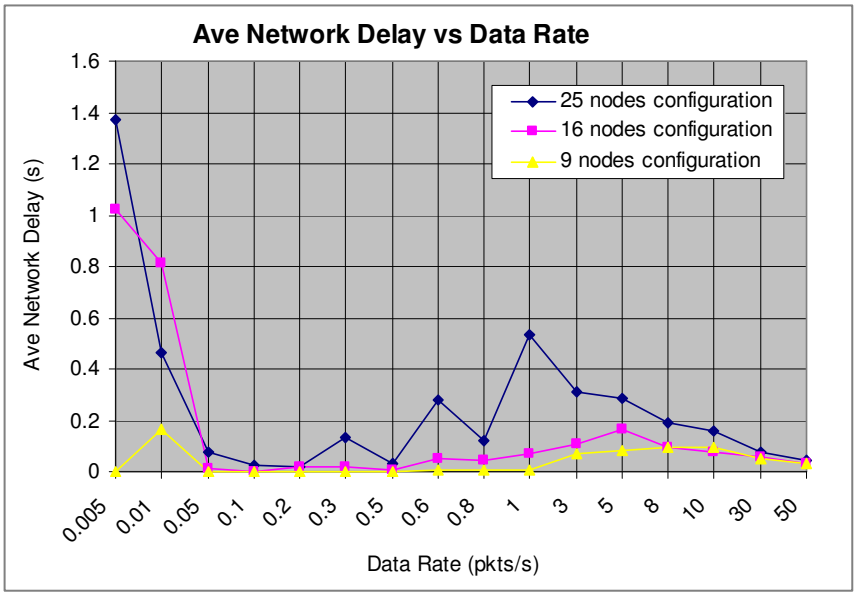


Figure 28 - Results of Average Network Delay with Varying Data Rate

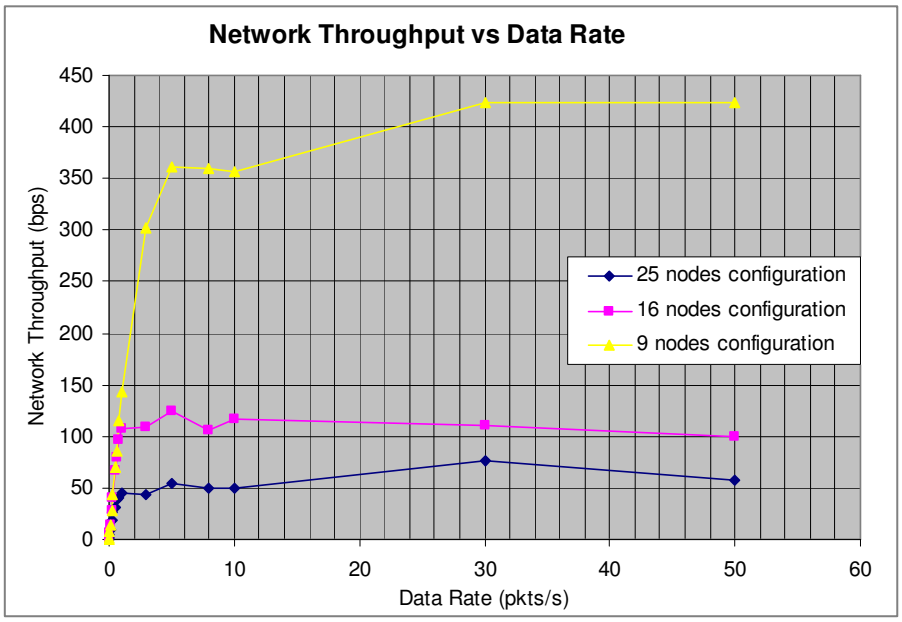


Figure 29 - Results of Network Throughput with Varying Data Rate

Figure 29 highlights the throughput analysis of the network. The impact on the throughput is apparent with different node configurations. It is observed that the network throughput increases linearly as the data rate is increased till a certain throughput for each configuration. Subsequently, the throughput remains almost

stagnant (gradual increase), possibly due to network congestion. The peak performance can be considered to be in the region of 5 to 8 packets per second. The 9-node configuration will have the highest throughput since all configurations has similar data rate but require less time for data to reach the destination.

4. Influence of Mobility

For the analysis of mobility effects on the network, the performance metrics are measured at various time intervals of the simulation and for the three node configurations. Since object movements, especially vessels or divers, in an ocean environment are relatively slow and do not change direction rapidly, a low data rate is considered. The movement of individual nodes will follow the path as computed from actual ocean data with the application agent transmitting at 0.01 packets per seconds, which is below the 0.3 packets per second data rate where the performance is deem acceptable.

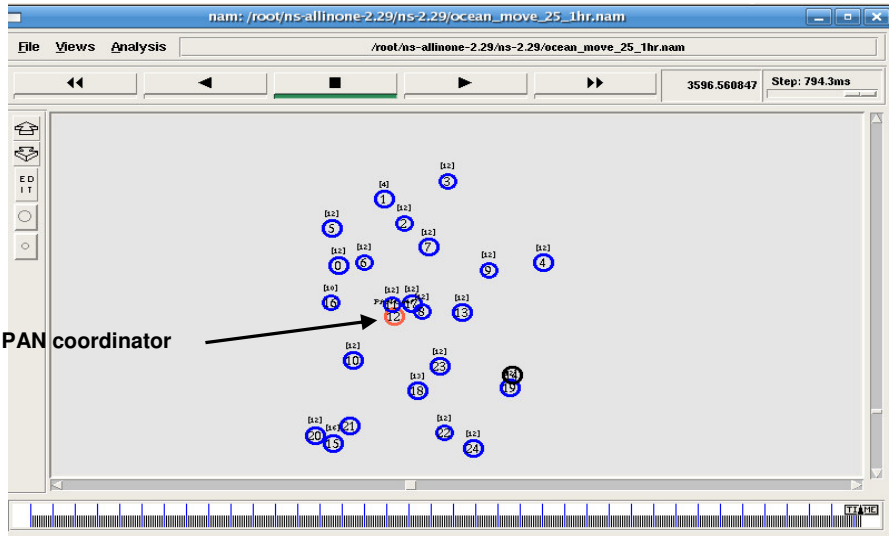


Figure 30 – Node Movements after 1 hour

To accommodate the node movements, a map size of 4500m × 4500m is considered for the simulations. The AODV routing protocol is applied to all

scenarios with the simulation runs varied at intervals in the region of 1 to 24 hours. An illustration of node mobility in the Network Animator (NAM) is shown in Figure 30 where the 25 nodes are dispersed after an hour of simulation. The node positions at various intervals of the three node configurations are as appended in Annex G.

The packet delivery ratio for this simulation is shown in Figure 31. The result shows that both the 25-node and 16-node configurations can achieve a PDR of approximately 80% and higher throughout the entire simulation. The 25-node configuration does not perform as well possibly due to more collisions that occur. Comparatively, the 9 nodes configuration has the worst results where the PDR reduces from above 80% till about 20% at the end of the 24 hours simulation. Intuitively, the PDR is expected to drop in the 9 nodes configuration since each node is required to provide a much larger coverage as the ocean current caused the nodes to be spread further apart.

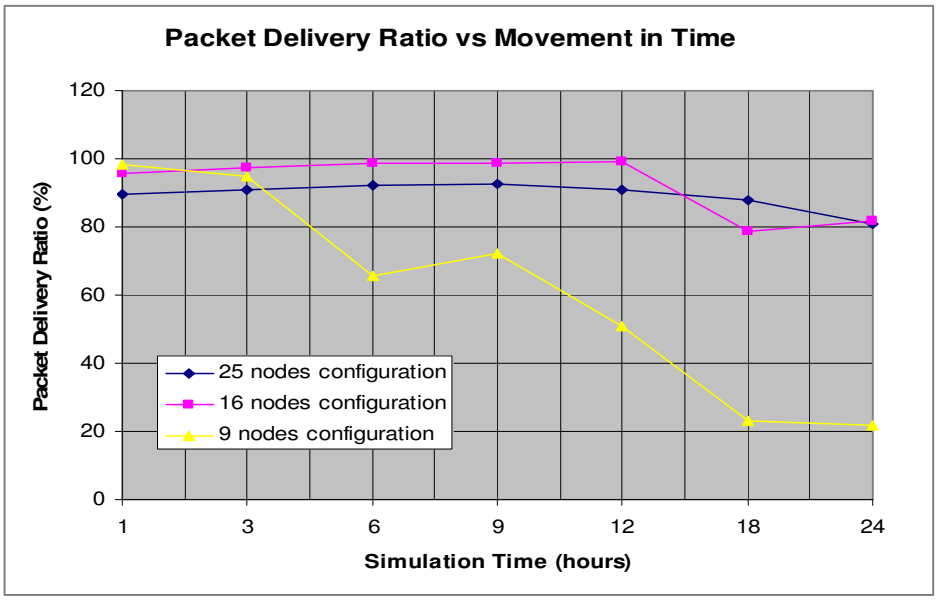


Figure 31 – Results of Packet Delivery Ratio with Node Movement in Time

Figure 32 shows the results for the network delay and throughput. In general, the network latency for the three configurations is below 0.1 seconds after the three hour simulation. The latency for the 9-node configuration is much lower since all nodes transmit directly to the PAN coordinator. An interesting observation is that in the first hour, the delay of network is comparatively higher in the 16-node configuration than the 25-node configuration. This behavior is attributed to the PAN coordinator being placed in an asymmetric position in the 16-node configuration as well as the distance between adjacent nodes.

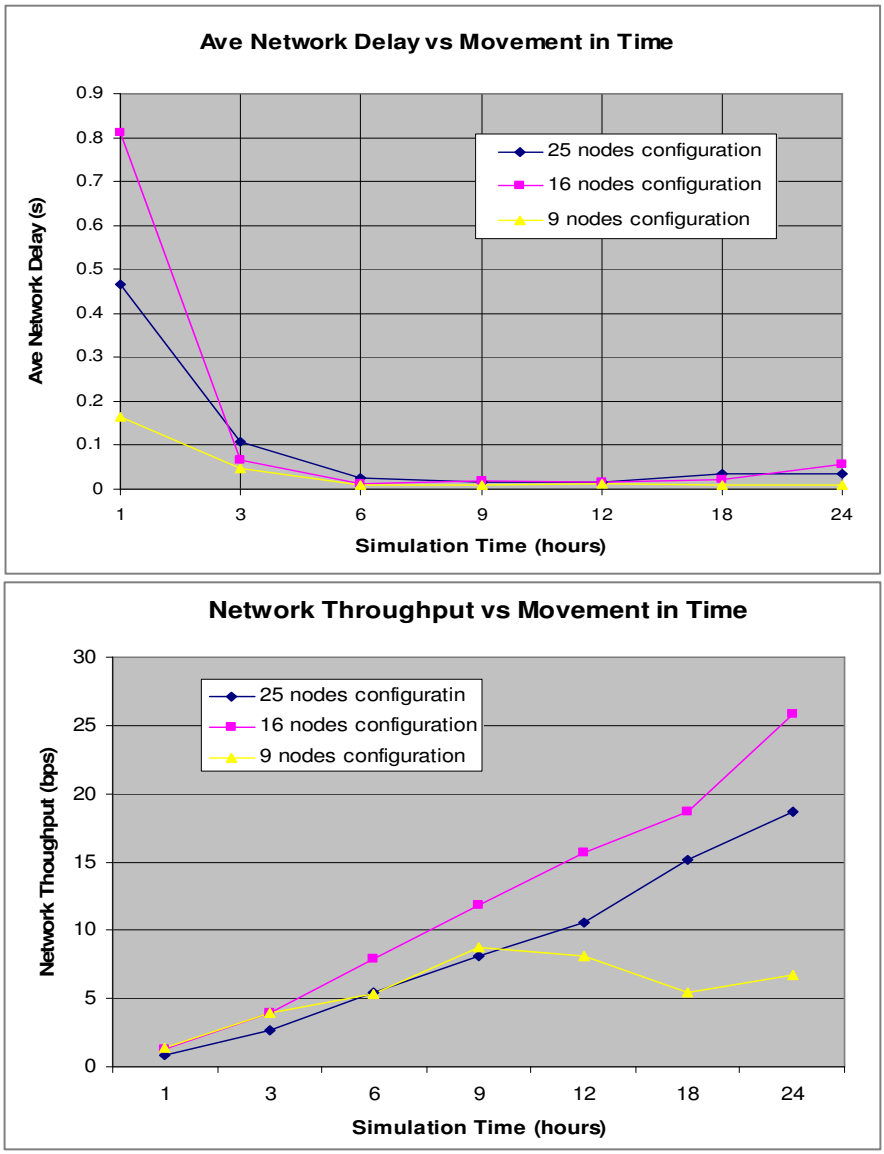


Figure 32 – Results of Network Delay & Throughput with Node Movement in Time

The network throughput in general, increases steadily over the entire simulation time. There is however a reduction in the throughput for the 9-node configuration scenario at the ninth hour. Since the trace files are too large to be analyzed, a probable reasoning can be deduced from the node positions. From Figure 33, the node positions of the 9-node configuration indicate that the PAN coordinator has drifted away from the other nodes in the ninth hour. Considering the sensing range, all the packets have to be routed through Node 1 to the PAN coordinator which results in the throughput reduction.

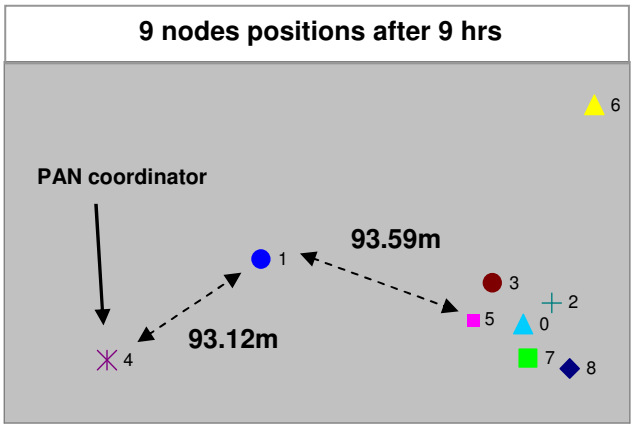


Figure 33 – Positions of 9 Nodes Configuration at the 9th Hour.

5. Drop Packet Analysis

As discussed in the above findings, packet drops will affect the system performance. Hence, an analysis is performed to study the likely cause for packets to drop. From certain trace files which are smaller in file size, it was observed there are two drop reasons that are of interest. In the network, packet drops are due to IFQ and LQI that occur frequently during data transmissions. IFQ is prompted when transmission rate exceeds the capacity of the queue. When that happens, the packets that join the queue first are serviced while the last is dropped (FIFO). A packet drop caused by LQI indicates that the link quality of packets arrived at the MAC layer is not within the acceptable threshold of the

CPThresh value. Such a drop is mainly caused when nodes have drifted apart. Hence, to reduce packet drops in the network, the queue length or the transmission range can be increased. Both queue length and transmission range is a hardware limitation determine by the buffer size and the transceiver on the device.

C. SUMMARY

In this chapter, simulations are conducted to analyze the performance of IEEE 802.15.4 in an ocean environment. The results based on the performance metrics of node density, network loading and mobility influences are discussed. In general, the network performance is very much affected by the node movements in the ocean. However, with the right selection of parameters, operating IEEE 802.15.4 is possible in ocean environment. However, these results cannot be applied in general as the mobility model only reflects the ocean condition at the Monterey Bay. In reality, the mobility of nodes will differ depending on the location. The next chapter concludes the analysis and the findings as well as recommendations for future possible research areas.

THIS PAGE INTENTIONALLY LEFT BLANK

V CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

A. CONCLUSIONS

This thesis is focused on determining the performance of IEEE 802.15.4 protocol in an ocean environment, in particular the surface current causing movement of floating nodes. A detailed study on the IEEE 802.15.4 architecture and the layered structure as well as the simulator environment has been conducted. All the simulations are conducted using open-source software, Network Simulator (NS2). Although mobility models are available in NS2, there are no suitable models to represent node movements in the ocean environment. Hence, actual measured ocean data of the Monterey Bay area is used as a basis for computing the movements. As the behavior of surface current differs from locations, the results from the analysis can only be representative of the Monterey Bay area.

For the purpose of conducting a near realistic simulation, the Zigbee transceiver parameters used are based on the specification of Crossbow MICAz processor and radio platform (MPR2400CA) which operates at 2.4 GHz. All the parameters are then incorporated into the TCL scripts. A detailed performance study is performed to determine the performance of IEEE 802.15.4, in particular to the performance metrics. The simulation results have indicated that the network performance is very much affected by the node movement in the ocean. Also, such technology can be successfully deployed for low data rate applications. However, by selecting an appropriate node configuration and network parameters such as data rate and transmission power, the network performance can be further optimized.

B. FUTURE RESEARCH AREAS

The results from the simulation have provided some insights into the use of IEEE 802.15.4/Zigbee wireless technology in an oceanic environment. Due to the time constraint and learning curve for NS2, certain simulation parameters are assumed and several features have not been studied. Hence, there are still other open research problems that are not addressed and also the simulation parameters in NS2 can still be further improved to represent the actual environment and deployed devices for more realistic result. The following details some key areas (non-exhaustive) for future research that can be used in future thesis research work to either augment the results made in this thesis or conducted in an independent thesis work.

1. Vertical Node Movement in Oceanic Environment

The current ocean model used in the simulation assumes a flat topology and only considers the horizontal movement of the nodes. In the actual ocean environment, the waves could cause lateral movements that result in intermittent connection due to the line-of-sight between nodes. A 3D mobility model can be created using the setdest function by defining the z parameter for the nodes.

2. Propagation Model

There is a need to refine the two-ray ground model used in the simulations to represent the ocean environment. The ocean is a harsh environment for propagating radio waves due to wave blockage, scattering and reflection of signals by the ocean surface causing multipath propagation which results in signal fading or signal loss. The Advanced Refractive Effects Prediction System (AREPS) model [36] is currently in use by the Navy and can be used to accurate

model the RF propagation on the ocean surface. The APEPS model takes into account factors that affect radio wave propagation in the maritime environment, including atmospheric refraction and surface reflections.

3. Node Energy Consumption

One of IEEE 802.15.4's features is the low-power consumption property. Hence, power consumption of the node is vital which provides another dimension to measure the performance. This can be accomplished by including an energy model into the simulation and also implementing or modifying existing codes to include low power mode. The energy consumption can then be investigated at the mobile nodes.

4. Security Considerations for Network

There are certain security risks in utilizing IEEE 802.15.4 such as eavesdropping, packet injection, replay, jamming and application vulnerabilities. Some of these risks can be eliminated with three security options in IEEE 802.15.4. The confidentiality aspect is addressed through the use a stream cipher encryption while integrity is addressed using a message integrity code (MIC) to protect data from being modified without the appropriate cryptographic key. The replay protection is implemented with the use of sequence number. Although the inclusion of these security mechanisms is necessary, the impact of the network performance has to be assessed.

5. Guaranteed Time Slot Management

GTS implementation is particularly effective for application that has timing constraints where each device is allocated specific time duration for high priority communication. The performance of IEEE 802.15.4 guaranteed time slot (GTS)

management scheme is not investigated in this thesis. Performance such as bandwidth utilization and data transfer reliability can be investigated.

APPENDIX A – INSTALLATION OF NS2

The NS2 is designed to operate on UNIX based operating system but it is also possible to run NS2 on Windows machine using Cygwin or on a VMware server. The following detailed the step for installation.

1. Download the all-in-one package (ns-allinone-2.29.tar.gz) from http://www.isi.edu/nsnam/ns/ns-build.html#allinone. The package contains the necessary components as well as an install script.

2. A new directory is created ("*mk ns*") and the compressed file is extracted into the new directory ("*tar -xzf ns-allinone-2.29.tar.gz*"). After extraction, change directory into ns-allinone-2.29 and run "*./install*". The install script will automatically configure, compile and install the components. If the installation is successful, the below message will be received.

```
Please put /root/ns/ns-allinone-2.29/bin:/root/ns/ns-allinone-
2.29/tcl8.4.13/unix:/root/ns/ns-allinone-2.29/tk8.4.13/unix into your
PATH environment; so that you'll be able to run
itm/tclsh/wish/xgraph.
.
.
(1) You MUST put /root/ns/ns-allinone-2.30/otcl-1.12, /root/ns/ns-
    allinone-2.30/lib, into your LD_LIBRARY_PATH environment
    variable.
    If it complains about X libraries, add path to your X libraries
    into LD_LIBRARY_PATH.
    If you are using csh, you can set it like:
            setenv LD_LIBRARY_PATH <paths>
    If you are using sh, you can set it like:
            export LD_LIBRARY_PATH=<paths>

(2) You MUST put /root/ns/ns-allinone-2.29/tcl8.4.13/library into
your TCL_LIBRARY environmental variable. Otherwise ns/nam will
complain during startup.
```

3. Navigate back to the root directory and there will be a .bashrc file. This is the file to place the necessary path environment and a backup of this file is created to allow restoration to original state if anything fails ("*cp .bashrc*

.bashrc.origin"). Environment variables are variables that Unix keeps track of at the shell level.

4. The .bashru file can be edited using gedit program and including the following script (directory can be different depending on where ns2 is installed) at the end of the file. With the current console window still open, go to another console window and determining the contents of environment variables by issuing the echo command.

```
PATH=/root/ns/ns-allinone-2.29/bin:/root/ns/ns-allinone-
2.29/tcl8.4.13 /unix:/root/ns/ns-allinone-2.29/tk8.4.13/unix:$PATH

LD_LIBRARY_PATH=/root/ns/ns-allinone-2.29/otcl-1.12:/root/ns/ns-
allinone-2.29/lib:$LD_LIBRARY_PATH

TCL_LIBRARY=/root/ns/ns-allinone-2.29/tcl8.4.13/library:$TCL_LIBRARY
```

The echo command in each case (i.e. $PATH, $LD_LIBRARY_PATH and $TCL_LIBRARY) will return the PATH variable that contains a list of previously included pathnames that is colon-delimited.

5. Finally, to verify that ns2 has been successfully installed, enter "*which ns"* which will return /root/ns/ns-allinaone-2.29/bin/ns and "ns" which will return the ns command prompt "%" (enter *"exit"* on the prompt to exit ns). NS2 can then be validate at the ns-2.29 directory with *"./validate"* command.

APPENDIX B – MOVEMENT PATTERN SCRIPT

The script below is modified from a sample script provided by Matthias Budde in the NS forum, customized to our computed data. This script essentially parses the numerous data generated by the node movement computation using the Tclsh command. Data points are extracted and arranged in the required format of the movement file. The computed movement data have to be arranged in anti-chronological order for subsequent computation of GOD object.

```
### SCRIPT TO PARSE DATA INPUT AND GENERATE MOVEMENT EVENT FOR NS-2
### Matthias Budde 2006, edited by Lim, Kwang Yong


set infilename    "./Nodemove2.txt"
set outfilename "./oceanmovement2.scn"

set infile [open $infilename ]
set outfile [open $outfilename w]


#Procedure to calculate seconds from time
proc gettime {hrs mins} {
      puts -nonewline "Transforming time $hrs:$mins to seconds: "
      set time [expr ($hrs * 3600) + ($mins * 60)]
      puts "$time"
      return $time
}

#Main

puts ""
puts "Parsing file \"$infilename\""
puts ""

while { [ gets $infile line ] != -1 } {

# This regular expression looks for a line formatted like
#"N:N   N.N   N.N" with n being any amount of digits from
#0 to 9 and saving them to the variables a to d

      if    {[regexp    {([0-9]+):([0-9]+)[^0-9]+([-0-9]+.[0-9]+)[^0-
9]+([-0-9]+\.[0-9]+)} $line z a b c d]} {

            set newtime [gettime $a $b]
            set newx $c
            set newy $d
```

```
if {$newtime == 0} {

                puts "Setting initial Position for Node $node"
                puts $outfile "$node set X_ $newx"
                puts $outfile "$node set Y_ $newy"
                puts $outfile "$node set Z_ 0.0"
          }

          set oldx $newx
          set oldy $newy

# in the elseif, the node ID is changed if the line doesn't look
like above but has the Word "Node " followed by a number. Also the
time is reset to 0
     } elseif {[regexp {Node *([0-9]+)} $line z a]} {

          set nodeID [expr $a - 1]
          puts "Setting Node ID to $nodeID"
          set node "\$node_($nodeID)"
          set newtime 0
# in the else, if the line looks any other than the two above, it's
ignored
     } else {
     # do nothing
     }

}

# This while statement reads the inputfile line per line

set infile [open $infilename ]

while { [ gets $infile line ] != -1 } {

# This regular expression looks for a line formatted like
#"N:N   N.N   N.N" with n being any amount of digits from
#0 to 9 and saving them to the variables a to d

     if   {[regexp   {([0-9]+):([0-9]+)[^0-9]+([0-9]+\.[0-9]+)[^0-
9]+([0-9]+\.[0-9]+)[^0-9]+(.|[0-9]+\.[0-9]+)} $line z a b c d e ]} {

          set newtime [gettime $a $b]
          set newx $c
          set newy $d

          if {$newtime > 0} {

                set speed $e
                puts $outfile "\$ns_ at $newtime.0 \"$node setdest
$newx $newy $speed\""

          }
```

```
# in the elseif, the node ID is changed if the line doesn't look
like above but has the Word "Node " followed by a number. Also the
time is reset to 0
      } elseif {[regexp {Node *([0-9]+)} $line z a]} {

            set nodeID [expr $a - 1]
            puts "Setting Node ID to $nodeID"
            set node "\$node_($nodeID)"
            set newtime 0
# in the else, if the line looks any other than the two above, it's
ignored
      } else {
      # do nothing
      }

}
      puts ""
      puts "DONE. Output has been written to \"$outfilename\". "
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C – COMMUNICATION PATTERN SCRIPT

The traffic connection pattern script is modified from a modification (Vaddina Prakash Rao) of the original constant bit rate generation script file by University of Southern California that is specifically designed for a star topology. The script is modified to generate traffic at an interval of 10 minutes which coincide with node movements. Nodes will be randomly selected to transmit data to the PAN coordinator.

```
#The copyright for the original script lies with the following author.
#Copyright (c) 1999 by the University of Southern California
#All rights reserved.
#Previous modified by Vaddina Prakash Rao

#Modified by:
#Lim Kwang Yong

#Traffic source generator from CMU's mobile code.

#Program to generate random traffic sources from nodes to coordinator.
# Traffic is generated every ten minutes customized to ocean node movement.

# ======================================================================
# Default Script Options, following default values are used when not provided
# ======================================================================

set opt(nn)             0           ;# Number of Nodes
set opt(seed)           0.0         ;# Random number generator seed value
set opt(mc)             0           ;# Number of sources
set opt(pktsize)        70          ;# Packet size

set opt(rate)           0           ;# Datarate
set opt(interval)       0.0         ;# inverse of rate
set opt(type)           ""          ;# Traffic type
set opt(starttime)      20.0        ;# Traffic start time (need WPAN to
establish)
set opt(endtime)        86400.0     ;# Traffic end time (1 day)
set opt(timegap)        30.0        ;# Default traffic gap time

# ======================================================================

set opt(outfilename) "./cbr/trafficpattern.traff"
set opt(outfile) [open $opt(outfilename) w]
```

```
#=============================================
# Procedure telling users the optional parameters
#=============================================
proc usage {} {
    global argv0

    puts "\nusage: $argv0 \[-type cbr|tcp\] \[-nn nodes\] \[-seed seed\]
\[-mc connections\] \[-rate rate\] \[-starttime st\] \[-endtime et\] \[-
timegap tg\]\n"
}

#====================================
# Procedure to get optional parameters
#====================================
proc getopt {argc argv} {
      global opt
      lappend optlist nn seed mc rate type starttime timegap

      for {set i 0} {$i < $argc} {incr i} {
            set arg [lindex $argv $i]
            if {[string range $arg 0 0] != "-"} continue

            set name [string range $arg 1 end]
            set opt($name) [lindex $argv [expr $i+1]]
      }
}

#==============
# CBR Procedure
#==============
proc create-cbr-connection { src dst stime } {
      global rng cbr_cnt opt
      puts $opt(outfile) "#\n# $src connecting to $dst at time $stime\n#"
      puts $opt(outfile) "set udp_($cbr_cnt) \[new Agent/UDP\]"
      puts $opt(outfile) "\$ns_ attach-agent \$node_($src)
\$udp_($cbr_cnt)"
      puts $opt(outfile) "set null_($cbr_cnt) \[new Agent/Null\]"
      puts $opt(outfile) "\$ns_ attach-agent \$node_($dst)
\$null_($cbr_cnt)"
      puts $opt(outfile) "set cbr_($cbr_cnt) \[new
Application/Traffic/CBR\]"
      puts $opt(outfile) "\$cbr_($cbr_cnt) set packetSize_ $opt(pktsize)"
      puts $opt(outfile) "\$cbr_($cbr_cnt) set interval_ $opt(interval)"
      puts $opt(outfile) "\$cbr_($cbr_cnt) set random_ 1"
      puts $opt(outfile) "\$cbr_($cbr_cnt) set maxpkts_ 10000"
      puts $opt(outfile) "\$cbr_($cbr_cnt) attach-agent \$udp_($cbr_cnt)"
      puts $opt(outfile) "\$ns_ connect \$udp_($cbr_cnt)
\$null_($cbr_cnt)"

      puts $opt(outfile) "\$ns_ at $stime \"\$cbr_($cbr_cnt) start\""

      incr cbr_cnt
}
```

```
#==============
# TCP Procedure
#==============

proc create-tcp-connection { src dst stime } {
      global rng cbr_cnt opt
      puts $opt(outfile) "#\n# $src connecting to $dst at time $stime\n#"
      puts $opt(outfile) "set tcp_($cbr_cnt) \[\$ns_ create-connection \
            TCP \$node_($src) TCPSink \$node_($dst) 0\]";
      puts $opt(outfile) "\$tcp_($cbr_cnt) set window_ 32"
      puts $opt(outfile) "\$tcp_($cbr_cnt) set packetSize_ $opt(pktsize)"
      puts $opt(outfile) "set ftp_($cbr_cnt) \[\$tcp_($cbr_cnt) attach-
source FTP\]"
      puts $opt(outfile) "\$ns_ at $stime \"\$ftp_($cbr_cnt) start\""

      incr cbr_cnt
}

#=============
# Main Program
#=============

getopt $argc $argv

if { $opt(type) == "" } {
    usage
    exit
} elseif { $opt(type) == "cbr" } {
    if { $opt(nn) == 0 || $opt(seed) == 0.0 || $opt(mc) == 0 || $opt(rate)
== 0 } {
      usage
      exit
    }

    set opt(interval) [expr 1.0 / $opt(rate)]
    if { $opt(interval) <= 0.0 } {
      puts "\ninvalid sending rate $opt(rate)\n"
      exit
    }
}

puts "#\n# nodes: $opt(nn), max conn: $opt(mc), send rate: $opt(interval),
seed: $opt(seed)\n#"

# Loop to transmit traffic within defined interval
set looptime [expr $opt(endtime)/600]
for {set k 1} {$k < $looptime} {incr k} {


set rng [new RNG]
$rng seed $opt(seed)
```

```
set u [new RandomVariable/Uniform]
$u set min_ 0
$u set max_ $opt(nn)
$u use-rng $rng

set cbr_cnt 0
set src_cnt 0
set flag 0
set loopcount $opt(nn)

# Loop till the defined connenction has completed, generated scr number
for {set i 0} {$i < $loopcount } {incr i} {

      while {1} {
      set x [$u value]
      set x [expr $x/0.01]
      if {$x < $opt(nn)} {
          set src [expr round($x)]
          break
      }
      }

      if { $src >= $opt(nn) } {
      set src [expr $opt(nn)-1]
      }

      # Add the generated source to the list unless we already had the
source before
      if {[array size sources] == 0} {
          set sources(0) $src
      } else {
      # Comparing each source in list to currently generated source to see
if match
          for {set p 0} {$p < [array size sources]} {incr p 1} {
            if {$src == $sources($p)} {
                set flag 1
                break
            }
          }

          if {$src == 12} {
            set flag 1
          }

# If there is a match, break from the current loop (discard the source)
          if {$flag == 1} {
            set flag 0
            incr loopcount 1
            continue
          } else {
            set size [array size sources]
            set sources($size) $src
          }
      }
```

```
      # Destination to be node-12 (PAN coordinator).
      set dst 12
      if { $opt(type) == "cbr" } {
            incr src_cnt 1
            create-cbr-connection $src $dst $opt(starttime)
      } else {
            create-tcp-connection $src $dst $opt(starttime)
      }

      if { $src_cnt == $opt(mc) } {
      # You have created the required number of sources. So break now !!
            break
      } else {
      # If required number of sources have not been reached yet.
          set temp $loopcount
          incr temp -1
          if {$i == $temp} {
            incr loopcount 1
          }
      }
      # Tx each node at an interval
      set opt(starttime) [expr $opt(starttime)+$opt(timegap)]
}

# Tx in the next cycle and reset source array
set opt(starttime) [expr $k*600]
unset sources

}

puts $opt(outfile) "#\n#Total sources: $src_cnt\n#"
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D – SAMPLE TCL SCRIPT

The sample TCL script provided is the configuration file for simulation. Some of the parameters are hard coded which needs to be varied and monitored to execute different sets of experimentations.

```
            ##############################################
            #         802.15.4 (beacon enabled)          #
            #      Copyright (c) 2003 Samsung/CUNY       #
            # - - - - - - - - - - - - - - - - - - - - - - #
            #        Prepared by Jianliang Zheng         #
            #         (zheng@ee.ccny.cuny.edu)           #
            #                                            #
            #      Modified By: Lim Kwang Yong           #
            #      Naval Post-Graduate School            #
            ##############################################

# Ocean movement scenario 25 nodes, 10m apart from adjacent nodes
# ======================================================================
# Set parameters regarding channel, medium, MAC protocol, routing
protocol, energy model ...
# ======================================================================
set val(chan)     Channel/WirelessChannel       ;# Channel Type
set val(prop)     Propagation/TwoRayGround      ;# radio-propagation model
set val(netif)    Phy/WirelessPhy/802_15_4
set val(mac)      Mac/802_15_4
set val(ifq)      Queue/DropTail/PriQueue       ;# interface queue type
set val(ll)       LL                            ;# link layer type
set val(ant)      Antenna/OmniAntenna           ;# antenna model
set val(ifqlen)   150                           ;# packet in ifq
set val(nn)       25                            ;# number of mobilenodes
set val(rp)       AODV                          ;# routing protocol
set val(x)        4500                          ;# x dimension topography
set val(y)        4500                          ;# y dimension topography
set val(tr)       ocean_move_25_3hr.tr          ;# trace file
set val(nam)      ocean_move_25_3hr.nam         ;# nam trace file
set val(move)     oceanmove3hr.scn              ;# movement scenario
set val(cp)       traffic_3hr_pattern.traff     ;# connection traffic
#set val(Energy)  EnergyModel                   ;# Evaluate energy

set val(starttime)      5                           ;
set val(appTime1)       [expr $val(starttime)+20]   ;
set appTime2            [expr $val(appTime1)+0.3]   ;
set stopTime            10800                       ;
```

```
#==========================
#read command line arguments
#==========================
proc getCmdArgu {argc argv} {
        global val
        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set val($name) [lindex $argv [expr $i+1]]
        }
}

getCmdArgu $argc $argv

#======================================
# Define antenna at node centre of ht 0.1m
#======================================
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 0.1
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

#============================
# Initialize Global Variables
#============================
set ns_           [new Simulator]

#=========================================================
# Create trace object for ns and nam, use new trace format
#=========================================================
set tracefd     [open ./$val(tr) w]
$ns_ trace-all $tracefd
# Statement indicating to simulator about NAM.
set namtrace     [open ./$val(nam) w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
#$ns_ use-newtrace

#=========================================================================
# Inform nam that this is a trace file for wpan (special handling needed)
#=========================================================================

$ns_ puts-nam-traceall {# nam4wpan #}

Mac/802_15_4 wpanCmd verbose on
Mac/802_15_4 wpanNam namStatus on         ;# default = off (should be
                                          turned on before other 'wpanNam'
                                          commands can work)
Mac/802_15_4 wpanNam ColFlashClr black    ;# default = gold
```

```
#========================
# For model 'TwoRayGround'
#========================
Phy/WirelessPhy set CSThresh_ 3.981e-13;   #carrier sensing threshold
Phy/WirelessPhy set RXThresh_ 3.981e-13;   #receiver threshold
Phy/WirelessPhy set CPThresh_ 10;          #capture threshold
Phy/WirelessPhy set freq_ 2.4e+9;          #Operating Freq
Phy/WirelessPhy set L_ 1.0;                #System loss factor
Phy/WirelessPhy set pt_ 0.001;             #Tx power

#===================================
# set up and define topography object
#===================================
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

#==========
#Create God
#==========
set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

#===============
# configure node
#===============

$ns_ node-config -adhocRouting $val(rp) \
              -llType $val(ll) \
              -macType $val(mac) \
              -ifqType $val(ifq) \
              -ifqLen $val(ifqlen) \
              -antType $val(ant) \
              -propType $val(prop) \
              -phyType $val(netif) \
              -topoInstance $topo \
              -agentTrace ON \
              -routerTrace ON \
              -macTrace ON \
              -movementTrace ON\
              -channel $chan_1_
              #=======================================
              # Parameter req. only when energy logging
              #=======================================
              #-energyModel EnergyModel \
                    #-initialEnergy 13000 \
                    #-rxPower 0.0648 \
                    #-txPower 0.0744 \
                    #-idlePower 0.00000552 \
```

```
#=============================================
# Creates specified node and attached to channel
#=============================================
for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
      $node_($i) random-motion 0          ;# disable random motion
}

#=========================='
# Define movement scenario
#=========================
puts "Loading movement pattern ..."
source $val(move)

#===============
# SSCS Interface
#===============
$ns_ at 0.0 "$node_(12) NodeLabel \"PAN Coor\""

# Cmd used to start a new PAN with corresponding node as PAN coordinator
# default <txBeacon=1> <beaconOrder=3> <SuperframeOrder=3>
$ns_ at 0.0 "$node_(12) sscs startPANCoord 0" ;# startPANCoord

# Cmd used to start a device
# default <isFFD=1> <assoPermit=1> <txBeacon=0> <BO=3> <SO=3>
$ns_ at 0.3 "$node_(7) sscs startCTDevice 1 1 1"
$ns_ at 1.3 "$node_(13) sscs startCTDevice 1 1 1"
$ns_ at 1.7 "$node_(17) sscs startCTDevice 1 1 1"
$ns_ at 2.3 "$node_(11) sscs startCTDevice 1 1 1"

$ns_ at 3.3 "$node_(8) sscs startCTDevice 1 1 1"
$ns_ at 3.5 "$node_(18) sscs startCTDevice 1 1 1"
$ns_ at 3.6 "$node_(16) sscs startCTDevice 1 1 1"
$ns_ at 3.8 "$node_(6) sscs startCTDevice 1 1 1"

$ns_ at 4.3 "$node_(2) sscs startCTDevice 1 1 1"
$ns_ at 4.5 "$node_(14) sscs startCTDevice 1 1 1"
$ns_ at 4.8 "$node_(22) sscs startCTDevice 1 1 1"
$ns_ at 5.1 "$node_(10) sscs startCTDevice 1 1 1"

$ns_ at 5.6 "$node_(3) sscs startCTDevice 1 1 1"
$ns_ at 5.8 "$node_(19) sscs startCTDevice 1 1 1"
$ns_ at 6.0 "$node_(21) sscs startCTDevice 1 1 1"
$ns_ at 6.3 "$node_(5) sscs startCTDevice 1 1 1"

$ns_ at 6.8 "$node_(4) sscs startCTDevice 1 1 1"
$ns_ at 7.0 "$node_(24) sscs startCTDevice 1 1 1"
$ns_ at 7.3 "$node_(20) sscs startCTDevice 1 1 1"
$ns_ at 7.7 "$node_(0) sscs startCTDevice 1 1 1"
```

```
$ns_ at 8.2 "$node_(1) sscs startCTDevice 1 1 1"
$ns_ at 8.5 "$node_(9) sscs startCTDevice 1 1 1"
$ns_ at 8.8 "$node_(23) sscs startCTDevice 1 1 1"
$ns_ at 9.1 "$node_(15) sscs startCTDevice 1 1 1"

#$ns_ at $appTime2 "$node_(12) sscs stopBeacon"

Mac/802_15_4 wpanNam PlaybackRate 3ms

$ns_ at $val(appTime1) "puts \"\nTransmitting data ...\n\""

#=================================
# Setup traffic flow between nodes
#=================================

puts "Loading traffic pattern ..."
source $val(cp)

# Mac/802_15_4 wpanNam FlowClr [-p <packet_type_name>] [-s <src>] [-d
<dst>] [-c <clrName>]

   Mac/802_15_4 wpanNam FlowClr -p AODV -c red
   Mac/802_15_4 wpanNam FlowClr -p ARP -c green
   Mac/802_15_4 wpanNam FlowClr -p MAC -c navy
   Mac/802_15_4 wpanNam FlowClr -p tcp -c blue
   Mac/802_15_4 wpanNam FlowClr -p ack -c cyan4

#=================================
# defines the node size (3) in nam
#=================================
for {set i 0} {$i < $val(nn)} {incr i} {
      $ns_ initial_node_pos $node_($i) 3
}

#=================================
# Tell nodes when the simulation ends
#=================================
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"NS EXITING...\n\""
$ns_ at $stopTime "$ns_ halt"
```

```
proc stop {} {
    global ns_ tracefd val(starttime) val env
    $ns_ flush-trace
    close $tracefd
    set hasDISPLAY 0
    foreach index [array names env] {
        puts "$index: $env($index)"
        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
                set hasDISPLAY 1
        }
    }
   if { ("$val(nam)" == "ocean_move_25_3hr.nam") && ("$hasDISPLAY" == "1")
} {
      exec nam ocean_move_25_3hr.nam

    }
}

puts "\nStarting Simulation..."

$ns_ run

#==============
# End of script
#==============
```

APPENDIX E – SAMPLE TRACE FILE FORMAT

```
s 0.000640000 _12_ MAC  --- 0 CM7 8 [0 ffffffff c 0]
s 0.140160000 _12_ MAC  --- 0 CM7 8 [0 ffffffff c 0]
s 0.280320000 _12_ MAC  --- 0 CM7 8 [0 ffffffff c 0]
s 0.300960000 _7_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
s 0.564960000 _7_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
s 0.828640000 _7_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
s 1.301920000 _13_ MAC  --- 0 CM7 8 [0 ffffffff d 0]
r 1.302368026 _12_ MAC  --- 0 CM7 8 [0 ffffffff d 0]
s 1.304928026 _12_ MAC  --- 0 BCN 11 [0 d c 0]
s 1.564000000 _13_ MAC  --- 0 CM7 8 [0 ffffffff d 0]
s 1.702240000 _17_ MAC  --- 0 CM7 8 [0 ffffffff 11 0]
r 1.702688033 _12_ MAC  --- 0 CM7 8 [0 ffffffff 11 0]
s 1.703968033 _12_ MAC  --- 0 BCN 11 [0 11 c 0]
s 1.827360000 _13_ MAC  --- 0 CM7 8 [0 ffffffff d 0]
r 1.827808042 _7_ MAC  --- 0 CM7 8 [0 ffffffff d 0]
s 1.964640000 _17_ MAC  --- 0 CM7 8 [0 ffffffff 11 0]
s 2.091040000 _13_ MAC  --- 0 CM1 17 [0 c d 0]
r 2.091776026 _12_ MAC  --- 0 CM1 17 [0 c d 0]
s 2.091968026 _12_ MAC  --- 0 ACK 5 [0 d c 0]
r 2.092320052 _13_ MAC  --- 0 ACK 5 [0 d c 0]
s 2.092960000 _7_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
r 2.093408033 _12_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
s 2.095008033 _12_ MAC  --- 0 BCN 11 [0 7 c 0]
s 2.227680000 _17_ MAC  --- 0 CM7 8 [0 ffffffff 11 0]
s 2.301920000 _11_ MAC  --- 0 CM7 8 [0 ffffffff b 0]
r 2.302368026 _12_ MAC  --- 0 CM7 8 [0 ffffffff b 0]
D 2.302368043 _7_ MAC  APS 0 CM7 8 [0 ffffffff b 0]
s 2.303648026 _12_ MAC  --- 0 BCN 11 [0 b c 0]
s 2.355680000 _7_ MAC  --- 0 CM7 8 [0 ffffffff 7 0]
s 2.491680000 _17_ MAC  --- 0 CM1 17 [0 c 11 0]
r 2.492416033 _12_ MAC  --- 0 CM1 17 [0 c 11 0]
s 2.492608033 _12_ MAC  --- 0 ACK 5 [0 11 c 0]
r 2.492960067 _17_ MAC  --- 0 ACK 5 [0 11 c 0]
s 2.565280000 _11_ MAC  --- 0 CM7 8 [0 ffffffff b 0]
D 2.565728043 _7_ MAC  APS 0 CM7 8 [0 ffffffff b 0]

…

D 27.267808849 _2_ MAC  COO 0 BCN 12 [0 ffffffff 13 0]
D 27.267808860 _5_ MAC  COO 0 BCN 12 [0 ffffffff 13 0]
D 27.267808863 _1_ MAC  COO 0 BCN 12 [0 ffffffff 13 0]
D 27.267808880 _0_ MAC  COO 0 BCN 12 [0 ffffffff 13 0]
s 27.268180039 _18_ AGT  --- 2 cbr 20 [0 0 0 0] ------- [18:0 12:0 32 0]
[2] 0 1
r 27.268180039 _18_ RTR  --- 2 cbr 20 [0 0 0 0] ------- [18:0 12:0 32 0]
[2] 0 1
s 27.268180039 _18_ RTR  --- 2 cbr 40 [0 0 0 0] ------- [18:0 12:0 30 12]
[2] 0 1
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F – SAMPLE AWK SCRIPTS

This script is modified from the many example scripts provided in the NS2 forum to parse the data from in the trace file format.

```
# Lim Kwang Yong

# Usage: awk -f <awk_script.awk> <trace_file.tr>
# Parse a ns2 wireless trace file and generate performance metrics

BEGIN {

droppedRTGpackets=0;
droppedbRTGbytes=0;
droppedMACbytes=0;
droppedMACpackets=0;
droppedbDATAbytes=0;
droppedbDATApackets=0;
droppedIFQpackets=0;
droppedLQIbytes=0;
droppedLQIpackets=0;
droppedIFQbytes=0;
totalDROPPEDpackets=0;
totalDROPPEDbytes=0;
dataSent=0;
dataRecd=0;
sentData=0;
sentDataBytes=0;
TotalsentData=0;
TotalsentDataBytes=0;
recdData=0;
recdDataBytes=0;
sends=0;
recvs=0;
RTGbytes=0;
fwdData=0;
fwdDataBytes=0;
fwdRtg=0;
fwdRtgBytes=0;
fwdMac=0;
fwdMacBytes=0;
highest_packet_id=0; #used to compare with the packet id
highest_time=0
sum=0;
RTG=0;
MACDATA=0;
MACRTG=0;
}
```

```
{
#============================
# Check for the highest packet ID & time
#============================
action = $1;
time = $2;
node_id = $3
if ($7 =="cbr"){packet_id = $6;};
 if ( packet_id > highest_packet_id ) {highest_packet_id = packet_id;};
if ( time > highest_time) {highest_time = time}

#============
# Calculate delay
#============
if ( start_time[packet_id] == 0 ) {start_time[packet_id] = time;};
 if ( action != "D" ) {
      if ( action == "r" ) {end_time[packet_id] = time;};
 }
else {end_time[packet_id] = -1;};

#===========
#Calculate PDF
#===========
if (($1 == "s") && ($7 == "cbr") && ($4 =="AGT")) {
      sends++;
      dataSent=dataSent+$8;
}
if ( ( $1 == "r") &&  ( $7 == "cbr" ) && ( $4 == "AGT" ) ) {
      recvs++;
      dataRecd=dataRecd+$8;
}

#===============
# Calculate RTG load
#===============

if ((( $1 == "s") || ($1=="f") ) && ( $7 == "cbr" ) && ( $4 =="RTR" )) {
      sentData++;
      sentDataBytes=sentDataBytes+$8;
}
if ( ( $1 == "r") && ( $7 == "cbr" ) && ( $4 == "RTR" ) ) {
      recdData++;
      recdDataBytes=recdDataBytes+$8;
}

#==============
# RTG protocol pkts
#==============

if ( (($1=="f")  || ($1=="s") ) && ($4 == "RTR") && ($7 =="AODV" )  ) {
RTG++;
RTGbytes=RTGbytes+$8;
}
```

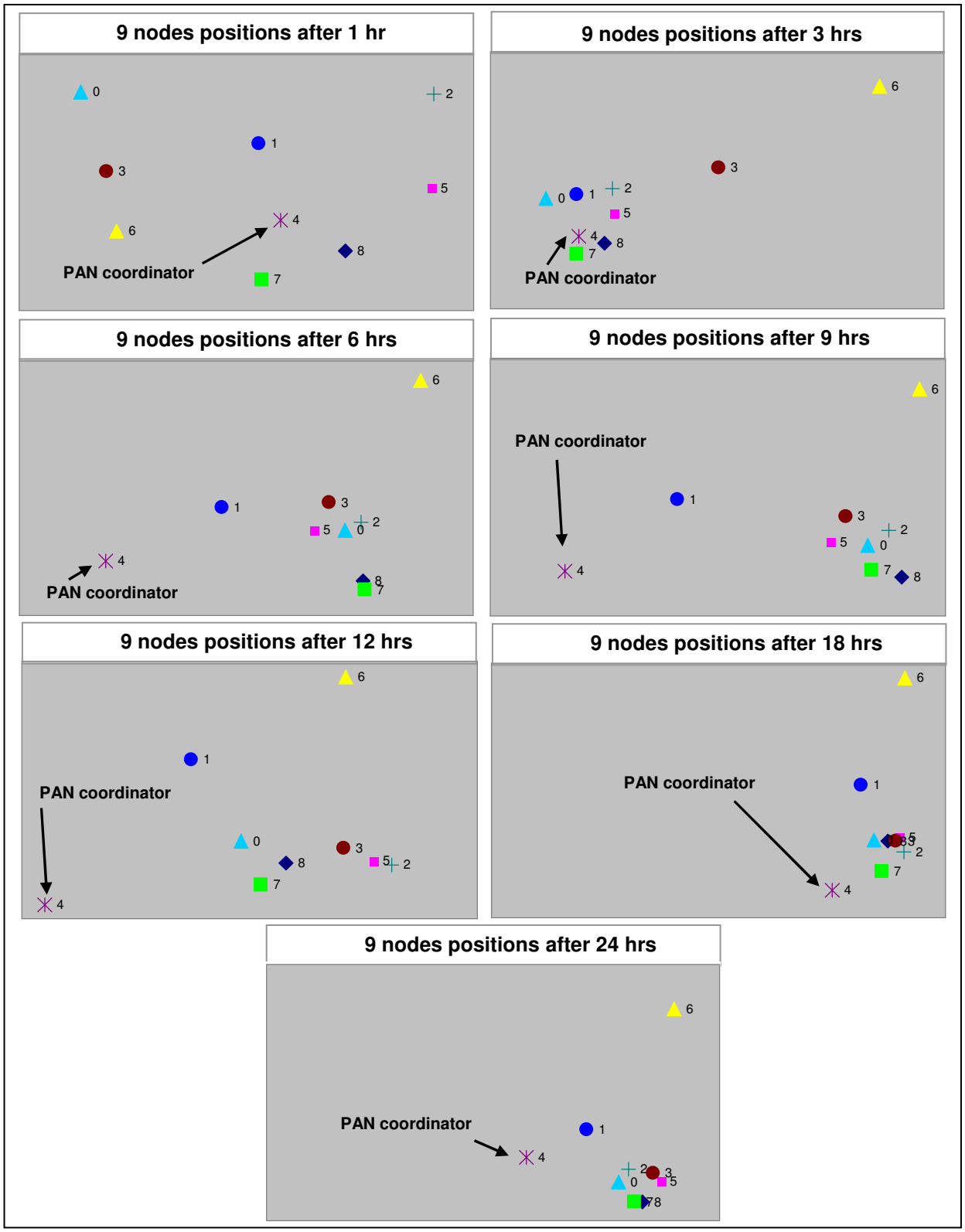
```
#============
# MAC data pkts
#============
if ( (($1=="f")  || ($1=="s") ) && ($4 == "MAC") && ($7 =="cbr" )  ) {
MACDATA++;
}

#=============
# MAC RTG  pkts
#=============
if ( (($1=="f")  || ($1=="s") ) && ($4 == "MAC") && ($7 =="AODV" )  ) {
MACRTG++;
}

#==============
# Dropped data pkts
#==============
if ( ($1 == "D") && ($7 =="cbr") && ( $4 == "RTR" )){
        droppedDATAbytes=droppedDATAbytes+$8 ;
        droppedDATApackets=droppedDATApackets+1;
}

#===============
# Dropped MAC pkts
#===============
if ( ($1 == "D") && ($4 == "MAC" )){
        droppedMACbytes=droppedMACbytes+$8 ;
        droppedMACpackets=droppedMACpackets+1;
}

#===============
# Dropped RTG pkts
#===============
if ( ($1 == "D") && ($4 == "RTR" ) && ($8 == "AODV")){
        droppedRTGbytes=droppedRTGbytes+$8 ;
        droppedRTGpackets=droppedRTGpackets+1;
}

#==============
# Dropped IFQ pkts
#==============
if (($1 == "D") && ($4 == "IFQ")){
        droppedIFQbytes=droppedIFQbytes+$8 ;
        droppedIFQpackets=droppedIFQpackets+1;
}

#==============
# Dropped LQI pkts
#==============
if (($1 == "D") && ($4 == "IFQ")){
        droppedLQIbytes=droppedLQIbytes+$8 ;
        droppedLQIpackets=droppedLQIpackets+1;
}
```

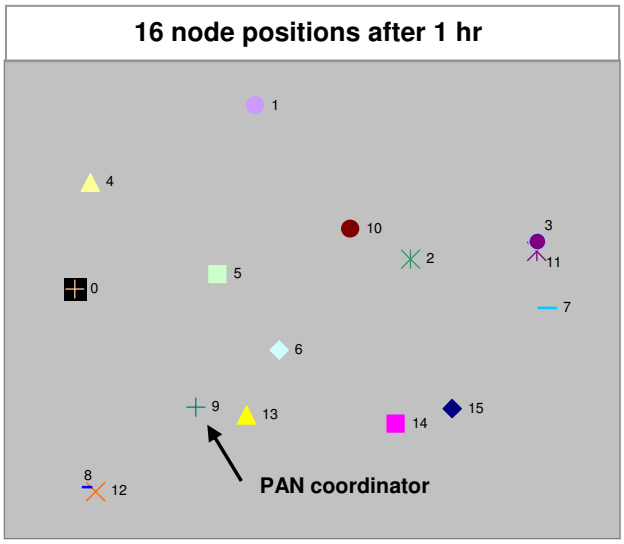
```
#==============
# Total dropped pkts
#==============
if ($1 == "D") {
        totalDROPPEDpackets++;
      totalDROPPEDbytes=totalDROPPEDbytes+$8
}

#==============
# Forward data pkts
#==============
if ( ($1 == "f") && ($7 == "cbr") && ($4 == "RTR" ) ) {
      fwdData++;
      fwdDataBytes=fwdDataBytes+$8;
}

#==============
# Forward RTG pkts
#==============
if ( ( $1 == "f") && ($7 == "AODV") && ($4 == "RTR") ) {
      fwdRtg++;
      fwdRtgBytes=fwdRtgBytes+$8;
}

#===============
# Forward MAC pkts
#===============
if ( ( $1 == "f") &&  ($4 == "MAC" ) ) {
      fwdMac++;
      fwdMacBytes=fwdMacBytes+$8;
}

#==============
# Total forward pkts
#==============
if  ( $1 == "f")  {
      fwdpackets++;
      fwdBytes=fwdBytes+$8;
}

if ($1 == "D") {
droppingnodes= $3;
}
}

END {

for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
       start = start_time[packet_id];
       end = end_time[packet_id];
       packet_duration = end - start;
   if (start < end) sum= packet_duration+sum;
}
```

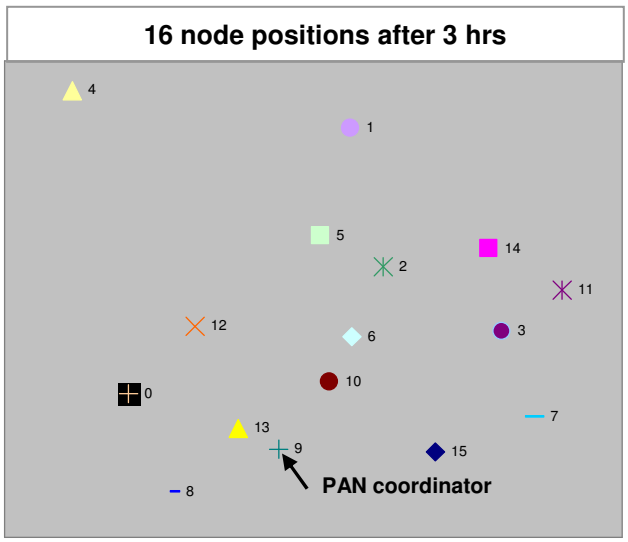
```
#==========
# Show results
#==========

printf("\n .......Data parsed from trace file with 25
NODES.............\n\n") >> "performance.txt";
printf(" 1.  No. of CBR Packets sent = %d PACKETS \n ", sends) >>
"performance.txt";
printf(" 2.  No. of CBR Bytes sent = %d BYTES \n", dataSent) >>
"performance.txt";
printf(" 3.  No. of CBR Packets received = %d PACKETS \n", recvs) >>
"performance.txt";
printf(" 4.  No. of CBR Bytes received = %d BYTES \n", dataRecd) >>
"performance.txt";
printf(" \n .......Routing details.............\n\n") >>
"performance.txt";
printf(" 5.  No. of routing packets = %d PACKETS \n",RTG) >>
"performance.txt";
printf(" 6.  No. of routing bytes = %d BYTES \n",RTGbytes) >>
"performance.txt";
printf(" \n .......MAC details.............\n\n") >> "performance.txt";
printf(" 7.  No. of MAC packets for data sent = %d PACKETS \n", MACDATA)
>> "performance.txt";
printf(" 8.  No. of MAC packets for Rtg sent  = %d PACKETS \n", MACRTG) >>
"performance.txt";
printf(" 9.  Total No. of MAC Packets sent    = %d PACKETS \n",
MACRTG+MACDATA) >> "performance.txt";
printf(" \n .......Forwarding details.............\n\n") >>
"performance.txt";
printf(" 10.  No. of DATA Packets Fwd = %d PACKETS \n", fwdData) >>
"performance.txt";
printf(" 11.  No. of DATA Bytes Fwd = %d BYTES \n", fwddataBytes) >>
"performance.txt";
printf(" 12.  No. of RTG Packets Fwd = %d PACKETS \n", fwdRtg) >>
"performance.txt";
printf(" 13.  No. of RTG Bytes Fwd = %d BYTES \n", fwdRtgBytes) >>
"performance.txt";
printf(" 14.  No. of MAC Packets Fwd = %d PACKETS \n", fwdMac) >>
"performance.txt";
printf(" 15.  No. of MAC Bytes Fwd = %d BYTES \n", fwdMacBytes) >>
"performance.txt";
printf(" 16.  Total No. of Fwd packets = %d PACKETS \n", fwdpackets) >>
"performance.txt";
printf(" 17.  Total No. of Fwd  Bytes = %d BYTES \n", fwdBytes) >>
"performance.txt";
printf(" \n ........Dropping details.............\n\n") >>
"performance.txt";
printf(" 18.  Dropping nodes are %d nodes \n", droppingnodes) >>
"performance.txt";
printf(" 19.  No. of dropped data (packets) = %d PACKETS
\n",droppedDATApackets) >> "performance.txt";
printf(" 20.  No. of dropped data (bytes) = %d BYTES \n",droppedDATAbytes)
>> "performance.txt";
```

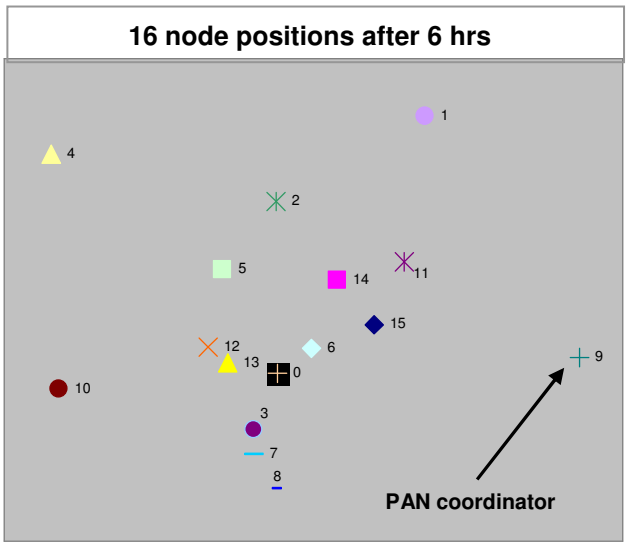
```
printf(" 21.  No. of dropped MAC (packets) = %d PACKETS
\n",droppedMACpackets) >> "performance.txt";
printf(" 22.  No. of dropped MAC (bytes) = %d BYTES \n",droppedMACAbytes)
>> "performance.txt";
printf(" 23.  No. of dropped RTG (packets) = %d PACKETS
\n",droppedRTGpackets) >> "performance.txt";
printf(" 24.  No. of dropped RTG (bytes) = %d BYTES \n",droppedRTGbytes)
>> "performance.txt";
printf(" 25.  No. of dropped IFQ (Packets) = %d PACKETS
\n",droppedIFQpackets) >> "performance.txt";
printf(" 26.  No. of dropped IFQ (bytes) = %d BYTES \n",droppedIFQbytes)
>> "performance.txt";
printf(" 27.  No. of dropped LQI (Packets) = %d PACKETS
\n",droppedLQIpackets) >> "performance.txt";
printf(" 28.  No. of dropped LQI (bytes) = %d BYTES \n",droppedIFQbytes)
>> "performance.txt";
printf(" 29.  Total No. of Dropped (packets) = %d PACKETS
\n\n",totalDROPPEDpackets) >> "performance.txt";
printf(" 30.  Total No. of Dropped (Bytes)) = %d BYTES
\n\n",totalDROPPEDbytes) >> "performance.txt";

printf("\n .............PERFORMANCE METRICES .....................\n\n")
>> "performance.txt";
printf(" 1.  Packet Delivery Ratio (PDR %) = %f %\n", (recvs/sends)*100)
>> "performance.txt";
printf(" 2.  Average Network delay = %f \n", sum/highest_packet_id) >>
"performance.txt";
printf(" 3.  Network Throughput = %f \n", (dataRecd/highest_time)/25) >>
"performance.txt";
printf(" 4.  Normalised routing load (Packets %)  = %f %\n",
(RTG/sentData)*100) >> "performance.txt";
printf(" 5.  Normalised routing load (Bytes %) =  %f %\n",
(RTGbytes/sentDataBytes)*100) >> "performance.txt";
printf(" 6.  Routing OVERHEAD (Packets %) = %f %\n", (MACRTG/MACDATA)*100)
>> "performance.txt";
printf(" 7.  Optimal Hop Count = %f \n\n", MACDATA/sends) >>
"performance.txt";
```

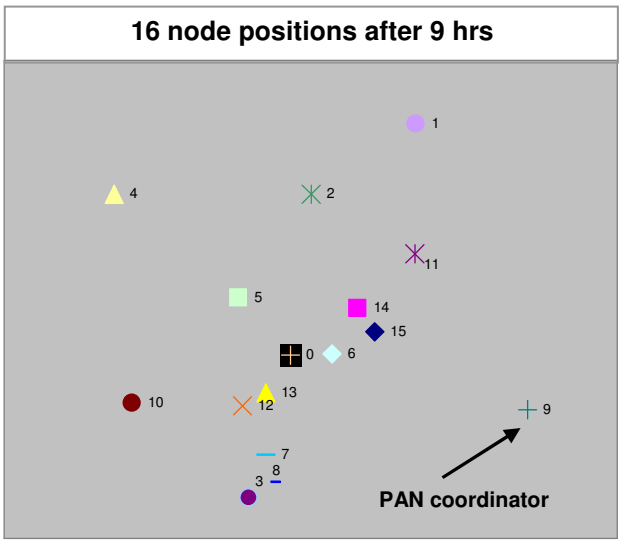
APPENDIX G – NODES POSITION AT VARIOUS TIME INTERVAL

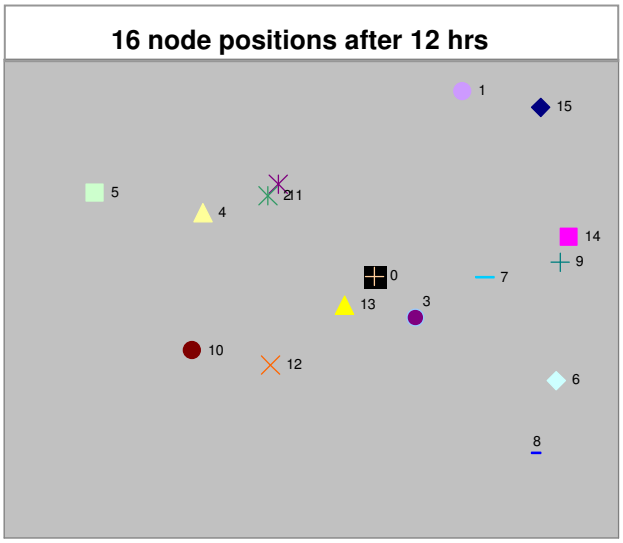



16 node positions after 1 hr
PAN coordinator

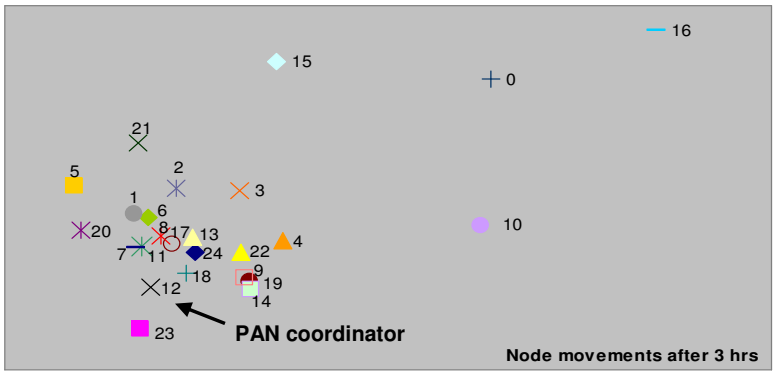

16 node positions after 3 hrs
PAN coordinator

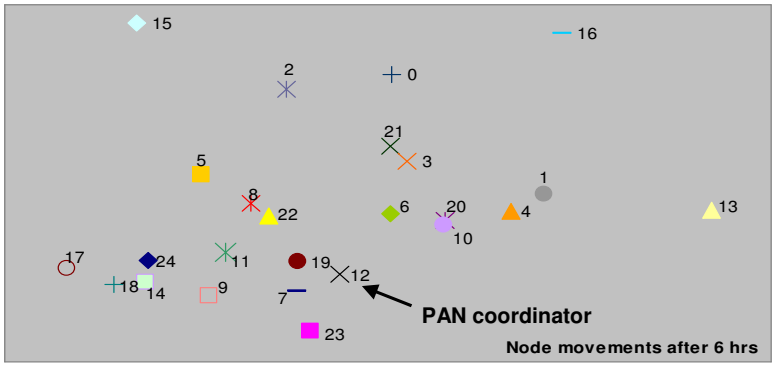

16 node positions after 6 hrs
PAN coordinator

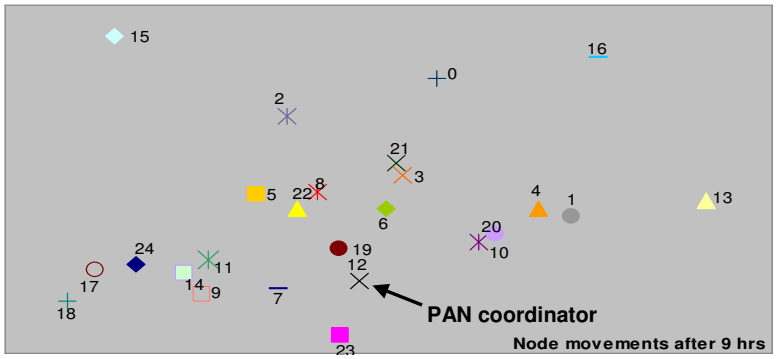

16 node positions after 9 hrs
PAN coordinator

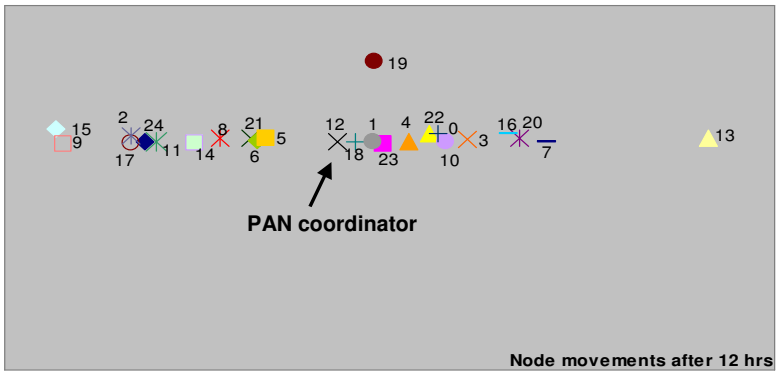

16 node positions after 12 hrs

25-node Configuration:









THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Wiki for Network Simulator (NS) and Network Animator (NAM), http://nsnam.isi.edu/nsnam/index.php/Main_Page, accessed on August 21, 2006.

2. Low Rate Wireless Personal Area Networks (LR-WPANs) – NS2 Simulation Platform, http://ees2cy.engr.ccny.cuny.edu/zheng/pub/, accessed on August 25, 2006.

3. J. Zheng and Myung J. Lee, "Will IEEE 802.15.4 make ubiquitous networking a reality? – a discussion on a potential low power, low bit rate standard," IEEE Communications Magazine, Vol. 42, No. 6, pp. 140-146, June 2004.

4. IEEE Computer Society, “IEEE Standard 802.15.4-2003 (draft), Part 15.4 : Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks”, IEEE Standards, October 2003.

5. J. Zheng and Myung J. Lee, "A comprehensive performance study of IEEE 802.15.4," *Sensor Network Operations*, IEEE Press, Wiley Interscience, Chapter 4, pp. 218-237, 2006.

6. José A. Gutierrez, Ed Callaway and Raymond Barrett, “Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4 “, IEEE Press, November 2003.

7. Joe Hoffert, Kevin Klues and Obi Orjih, “Configuring the IEEE 802.15.4 MAC Layer for Single-sink Wireless Sensor Network Applications”, Washington University in St. Louis.

8. ZigBee Alliance, “ZigBee Specification”, ZigBee Document 053474r06, Version 1.0, December 2004

9. Sinem Coleri Ergen, "ZigBee/IEEE 802.15.4 Summary", September 2004, http://www.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf

10. C. E. Perkins and E. M. Royer, “Ad Hoc On-demand Distance Vector Routing,” in Proc. 2nd IEEE Wksp. Mobile Comp. Sys. and Apps., February 1999, pp. 90-100.

11. IETF, “Ad hoc On-Demand Distance Vector (AODV) Routing”, RFC 3561, http://www.ietf.org/rfc/rfc3561.txt, assessed on September 27, 2006.

12. Surface and Subsurface Ocean Currents, http://www.geog.ouc.bc.ca/physgeog/fundamentals/8q.html, accessed on August 24, 2006.

13. Paul, R. Pinet, “Invitation to Oceanography, 3rd edition”, Jones and Bartlett Publishers Inc., 2003.

14. Ocean Currents, http://www.waterencyclopedia.com/Mi-Oc/Ocean-Currents.html, accessed on August 25, 2006.

15. Surface Ocean Current, www.nmm.ac.uk/server.php?show=conMediaFile.6569, accessed on August 25, 2006.

16. Ocean World, Currents - Ekman Spiral, http://oceanworld.tamu.edu/students/currents/currents3.htm), accessed on August 25, 2006.

17. Department of Earth Sciences, University of Southern California, Upwelling and Downwelling Effect, http://earth.usc.edu/~stott/Catalina/Oceans.html, accessed on August 25, 2006.

18. Tides and Water Levels, http://www.oceanservice.noaa.gov/education/kits/tides/tides01_intro.html, accessed on August 27, 2006.

19. Ross, David A., “Introduction to Oceanography”, New York: HarperCollins College Publishers, 1995.

20. National Oceanic and Atmospheric Administration, Tides and Water Levels, http://www.oceanservice.noaa.gov/education/kits/tides/tides01_intro.html, accessed on August 27, 2006.

21. Background (Mobility Models and Data for Wireless Networks), http://signl.cs.umass.edu/~liberato/wireless_background.html, accessed on August 28, 2006.

22. BonnMotion, A mobility scenario generation and analysis tool, http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/, accessed on August 11, 2006.

23. T. Camp, J Boleng, V. Davies, “A Survey of Mobility Models for Ad Hoc Network Research”, Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, 2002, vol. 2, no. 5, pp. 483-502.

24. B. Liang and Z. Haas, “Predictive distance-based mobility management for PCS networks”, Proceedings of the IEEE INFOCOM 1999, pp.1377-1384.

25. Laboratory S.A.M.O.V.A.R, Mobility model in ad hoc network, www-public.int-evry.fr/~gauthier/mobility.html, accessed on September 11, 2006.

26. “Selection Procedures for the Choice of Radio Transmission Technologies of the UMTS.” Technical Report tr 101 112 v3.2.0, ETSI SMG-5, France, April 1998.

27. X. Hong, M. Gerla, G. Pei and C.C. Chiang, “A Group Mobility Model for Ad Hoc Wireless Networks,” Proceeding of the 2nd ACM/IEEE Int. Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM'99), 1999, pp. 53-60.

28. REAL 5.0 Overview, http://www.cs.cornell.edu/skeshav/real/overview.html, accessed on October 22, 2006.

29. NS2 Event Scheduler, http://www.acims.arizona.edu/PUBLICATIONS /PDF/Thesis_Taekyu.pdf, accessed on October 22, 2006.

30. “The CMU Monarch Project’s Wireless and Mobility Extensions to *ns*”, The CMU Monarch Project, Computer Science Department, Carnegie Mellon University Snapshot Release 1.1.1, August 1999.

31. “The *ns* Manual (formerly *ns* Notes and Documentation)”, The VINT Project, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, August 2006.

32. NS-2 Trace Formats, http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats#New_Wireless_Trace_Formats, accessed on November 10, 2006.

33. Near Real-Time Monterey Bay Surface Currents, http://newark.cms.udel.edu/~brucel/realtimemaps/index.html, accessed on September 2, 2006

34. Ed Williams, Aviation Formulary V1.43, http://williams.best.vwh.net /avform.htm#LL, accessed on September 8, 2006.

35. Vaddina Prakash Roa, “Master Thesis: The simulative Investigation of Zigbee/IEEE 802.15.4”, Dresden University of Technology, November 2005.

36. Advanced Refractive Effects Prediction System (AREPS), http://sunspot.spawar.navy.mil/, accessed on December 1, 2006.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Peter Denning
   Chairman, Department of Computer Science
   Naval Postgraduate School
   Monterey, California

4. Professor John C. McEachen
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

5. Professor Gurminder Singh
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

6. Professor Yeo Tat Soon (email: tdshead@nus.edu.sg)
   Director, Temasek Defence Systems Institute (TDSI)
   National University of Singapore
   Singapore

7. Tan Lai Poh (email: tdstlp@nus.edu.sg)
   Temasek Defence Systems Institute (TDSI)
   National University of Singapore
   Singapore

8. Lim Kwang Yong
   Singapore Technologies Dynamic Pte Ltd
   Singapore